Fuzzy Logic Control of an Automated Guided Vehicle

Jeremy Baxter
B.Sc. (Dunelm)

School of Engineering
University of Durham

A thesis submitted in partial fulfilment of the requirements of the Council of the University of Durham for the Degree of Doctor of Philosophy (Ph.D.).

September 1994

Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

Abstract

This thesis describes the fuzzy logic based control system for an automated guided vehicle (AGV) designed to navigate from one position and orientation to another while avoiding obstacles.

A vehicle with an onboard computer system and a beacon based location system has been used to provide experimental confirmation of the methods proposed during this research. A simulation package has been written and used to test control techniques designed for the vehicle.

A series of navigation rules based upon the vehicle's current position relative to its goal produce a fuzzy fit vector, the entries in which represent the relative importance of sets defined over all the possible output steering angles. This fuzzy fit vector is operated on by a new technique called rule spreading which ensures that all possible outputs have some activation.

An obstacle avoidance controller operates from information about obstacles near to the vehicle. A method has been devised for generating obstacle avoidance sets depending on the size, shape and steering mechanism of a vehicle to enable their definition to accurately reflect the geometry and dynamic performance of the vehicle. Using a set of inhibitive rules the obstacle avoidance system compiles a mask vector which indicates the potential for a collision if each one of the possible output sets is chosen.

The fuzzy fit vector is multiplied with the mask vector to produce a combined fit vector representing the relative importance of the output sets considering the demands of both navigation and obstacle avoidance. This is operated on by a newly developed windowing technique which prevents any conflicts produced by this combination leading to an undesirable output. The final fit vector is then defuzzified to give a demand steering angle for the vehicle. A separate fuzzy controller produces a demand velocity.

In tests carried out in simulation and on the research vehicle it has been shown that the control system provides a successful guidance and obstacle avoidance scheme for an automated vehicle.

This thesis is dedicated to my wife, Rebecca.

Acknowledgments

I would like to acknowledge the assistance, support and encouragement provided by my supervisor Dr Jim Bumby.

Many thanks are also due to Peter Baxendale and Milos Kolar for their assistance in sorting out the interface between the transputers, programmable logic and electronics and to all in the mechanical and electronic workshops who helped assemble the vehicle.

Finally I would like to thank Dave Grey for letting me bounce ideas off him, John Glover for knowing far more about UNIX than I ever will and my wife, Rebecca Morgan, for always being there when I needed her.

Contents

Abstract i

Dedication ii

Acknowledgments iii

Contents iv

List of Figures x

List of Tables xv

Abbreviations xvi

1 Introduction and Review of Relevant Research 1

1.1 Vehicle Guidance Systems 2

1.2 Sensor Systems 3

1.2.1 Ultrasonic Transducers 3

1.2.2 Sonar Arrays 4

1.2.3 Infra Red Sensors 4

1.2.4 Lasers 4

1.2.5 Vision 5

1.2.6 Odometry 5

1.2.7 Other Sensors 5

1.3 Localisation 5

1.4 Mapping 6

1.5 Path Planning and Reactive Control 8

1.6 Local Obstacle Avoidance 9

1.7 Motion Control 10
1.8 Fuzzy Logic 11
1.9 Discussion of Relevant Literature 12
1.9.1 Planning versus Reaction 12
1.9.2 Mapping 14
1.9.3 Control Hierarchy 15
1.10 Outline of Research 16
1.11 Thesis Outline 19

2 AGV Hardware and Associated Sofware **21**
2.1 The Vehicle Hardware 22
2.1.1 The Vehicle 22
2.1.2 The Computer System 25
2.1.3 Digital Interface Electronics 26
2.2 Controlling Software Structure 27
2.2.1 Message Handler 29
2.2.2 Controlling Process 29
2.3 Motion Control 31
2.4 Equations of Motion 32
2.5 Velocity and Steering Angle Limits 34
2.6 Localisation System 37
2.6.1 Odometry 38
2.6.2 Beacon Location System 41
2.6.3 Location System Control 42

3 Simulation and Fuzzy Control Software **44**
3.1 The AGV Simulator 45
3.2 Initialisation Routines 47
3.2.1 Data Structures 48
3.2.2 Setup Routine 49
3.2.3 Navigation Rules Setup 49
3.2.4 Obstacle Avoidance Set Initialisation 50
3.2.5 Occupancy Grid Initialisation 50
3.3 Fuzzy Logic Software 51

3.4 Fuzzification Routines. 53
3.4.1 Input Fuzzification 53
3.4.2 Activation - Navigation Controller 53
3.4.3 Activation - Velocity Controller 55
3.5 Defuzzification Routines. 55
3.5.1 Standard Defuzzification 56
3.5.2 Windowing and Rule Spreading Routines 57
3.6 Motion Simulation Routine 58
3.7 Information Display 59

4 Fuzzy Navigation Controller **61**
4.1 Control System Design 61
4.1.1 State Variable Selection 62
4.1.2 Set Definitions 64
4.1.3 Rule Definition 69
4.2 Initial Problems 75
4.2.1 Conflicting Rules 75
4.2.2 Failure Caused by Dynamic Equilibrium 76
4.2.3 Obstacle Avoidance Considerations 76
4.3 Performance Testing of the Navigation Rulesets 79
4.3.1 Test 1 79
4.3.2 Test 2 84
4.3.3 Test 3 84
4.3.4 Test 4 89
4.3.5 Final Controller Docking Performance Tests 89
4.4 Velocity Effects on Navigation 93
4.5 Velocity Controller 94
4.6 Noise Effects on Navigation 103

5 Obstacle Avoidance Controller **105**
5.1 Obstacle Avoidance Principles 105
5.2 Obstacle Representation 106
5.3 The Form of Avoidance Rules 107
5.4 Definition of Avoidance Sets 108

5.5 Calculating the Activation of Avoidance Sets 111
5.6 Conflicting Outputs Can Produce Collisions 115
5.7 New Fuzzy Control Techniques 116
5.7.1 Sliding Window Defuzzification 117
5.7.2 Rule Spreading 118
5.8 Windowing and Rule Spreading Combined 119
5.9 Effects of the Obstacle Avoidance Controller 120
5.10 The Controller in Action. 122

6 Analysis and Development of the Avoidance Scheme 131
6.1 Dynamic Indecision 131
6.2 Demand Velocity Oscillations 134
6.3 Dynamic Windowing 137
6.4 Spreading Constant Values 140
6.5 Variable Rule Spreading 141
6.6 Geometric Obstacle Avoidance Set Generation 143
6.7 Comparison of Obstacle Avoidance Sets 148
6.7.1 Set Length 148
6.7.2 Minimal Sets 152
6.7.3 Set Width 154
6.7.4 Inhibition Values 155
6.7.5 The Final Avoidance Set Definition 157
6.8 Obstacle Avoidance Results 158

7 Performance of the Final Controller 162
7.1 Avoidance of Multiple Objects 163
7.2 Controller Sampling Frequency 165
7.3 Problem Situations 166
7.3.1 Wide Walls 166
7.3.2 Narrow Gaps 168
7.3.3 Dead Ends 169
7.4 A Potential Solution - Sub-goals 169
7.5 Longer Laboratory Tests. 173

8 Conclusions and Suggestions for Further Work **176**

8.1 Conclusions 176

8.2 The Contribution of the Presented Work 180

8.3 Suggestions for Further Work 180

A Fuzzy Logic Control **190**

A.1 Numerical Fuzzy Sets 191

A.2 Fuzzy Operators 192

A.3 Correlation 193

A.4 Defuzzification 195

A.5 The Structure of a Fuzzy Logic Controller 196

B Beacon Location System **198**

B.1 Triangulation 199

B.1.1 Triangulation Equations 199

B.1.2 Triangulation With Inaccurate Data 201

B.1.3 Position and Orientation Correction 203

B.2 Location System Hardware 205

B.2.1 Scanner 206

B.2.2 Receiver Electronics 207

B.3 Scanner Controlling Logic 209

B.3.1 Tracking Function 209

B.3.2 Other Functions 212

B.3.3 Event Generation 212

B.3.4 Computer Interface 213

B.4 Location System Software 214

B.4.1 Location Control Process 214

B.4.2 Beacon Assignment 217

B.4.3 Scanner Controlling Process 219

B.4.4 Scanner Driver Process 220

B.4.5 Synchronisation Process 222

C Memory Mapped I/O details **223**

C.1 Register Contents 225

D AGV Hardware **227**

D.1 Power Supplies and Isolation 227

List of Figures

1.1 Schematic Diagram of the Elements of a Vehicle Guidance System. 2
1.2 Layers in the Control Hierarchy 15
1.3 The Navigation Problem 16
1.4 Basic Control Loop for the Research Vehicle. 17

2.1 Plan View of the Research Vehicle 23
2.2 Side View of the Research Vehicle 24
2.3 Host PC and Onboard Computer 25
2.4 Subsystems Realised in Digital Logic 26
2.5 Primary Control Processes 28
2.6 Flow Chart of Actions Taken Every Sample Interval. 30
2.7 Block Diagram of Motion Control System 32
2.8 Diagram Showing Derivation of Turning Radius From Steering Angle 33
2.9 Maximum Rate of Change of Steering Constant Against Speed. 35
2.10 Limits Placed upon the Vehicle's Acceleration. 36
2.11 Change in Vehicle Position, Based on Odometric Measurements 39
2.12 Scanner Incorporating Infra Red Sensors 41
2.13 Location System Control Processes 42

3.1 Control Flow Through the Main Simulation Loop 46
3.2 Control Flow Through the Fuzzy Logic Routines 52
3.3 Visual Output from the Vehicle Simulator 60

4.1 The State Variables in Relation to the Global Coordinate System. 63
4.2 The Fuzzy Sets for the Output Steering Angle 64
4.3 Coarse Controller, Fuzzy Sets for the Distance to the Goal 65
4.4 Fine Controller, Fuzzy Sets for the Distance to the Goal 66

4.5 Final Controller, Fuzzy Sets for the Distance to the Goal 66
4.6 The Coarse Fuzzy Sets for the Goal and Heading Error Angles 67
4.7 Fuzzy Sets for the Goal and Heading Error Angles, Fine and Final Controllers 68
4.8 Vehicle Position for Set Activation Example 75
4.9 Example of Dynamic Equilibrium in a Faulty Controller Design 77
4.10 Controller Block Diagram, including Windowing and Rule Spreading 78
4.11 Path for Test 1, without Windowing and Rule Spreading. 80
4.12 Angle Error During Test 1, without Windowing and Rule Spreading. 81
4.13 Path for Test 1, using Windowing and Rule Spreading. 82
4.14 Angle Error During Test 1, using Windowing and Rule Spreading. 83
4.15 Path for Test 2. 85
4.16 Angle Error During Test 2. 86
4.17 Path for Test 3. 87
4.18 Angle Error During Test 3. 88
4.19 Path for Test 4. 90
4.20 Angle Error During Test 4. 91
4.21 Paths for Tests 5-8. 92
4.22 Vehicle Control Loop, Illustrating Rate Limiting Block. 93
4.23 Path for Test 1, Showing Velocity Effects. 95
4.24 Angle Error During Test 1, Showing Velocity Effects. 95
4.25 Path for Test 2, Showing Velocity Effects. 96
4.26 Angle Error During Test 2, Showing Velocity Effects. 96
4.27 Vehicle Controller Loop, Incorporating Velocity Control. 97
4.28 The Set Definitions used in the Velocity Controller 98
4.29 Demanded and Actual Velocity of AGV During Test 2 99
4.30 Demanded and Actual AGV Steering Angle During Test 2 99
4.31 Path for Test 1, Using Velocity Control. 101
4.32 Angle Error During Test 1, Using Velocity Control. 101
4.33 Path for Test 2, Using Velocity Control. 102
4.34 Angle Error During Test 2, Using Velocity Control. 102
4.35 Path for Test 3, Low Level Noise. 103
4.36 Path for Test 3, High Level Noise. 104

5.1 Avoidance Set Area for 'Negative Small' Steering Angle. 109
5.2 Distant Avoidance Set Areas for all Steering Angle Sets. 110
5.3 Avoidance Set Areas for 'Zero' Steering Angle. 111
5.4 Avoidance Set Area in Global and Local Co-ordinates. 112
5.5 Avoidance Set Area Mapped onto an Occupancy Grid. 113
5.6 Finding the Edges of a Set on the Grid. 114
5.7 Graph Illustrating Calculation of Acceptability. 115
5.8 Fuzzy Fit Vector Before and After Inhibition to Avoid a Possible Collision. 116
5.9 The Effect of Windowing On the Fuzzy Fit Vector. 118
5.10 Fuzzy Fit Vector Before and After Inhibition to Avoid a Possible Collision, with Windowing. 119
5.11 Rule Spreading Applied to a Fuzzy Fit Vector. 120
5.12 Block Diagram of Controller 121
5.13 Windowing Alleviating a Conflict in the Rules. 122
5.14 Vehicle in Laboratory Test Avoiding a Central Box. 123
5.15 Vehicle in Laboratory Test Avoiding a Central Box, Starting 1m to Left. . . 123
5.16 Vehicle Avoiding Two Obstacles. 124
5.17 Positions used for Analysis of Controller Behaviour. 124
5.18 Fuzzy Fit and Mask Vectors for Position 1. 126
5.19 Fuzzy Fit and Mask Vectors for Position 2. 128
5.20 Fuzzy Fit and Mask Vectors for Position 3. 129
5.21 Fuzzy Fit and Mask Vectors for Position 4. 130

6.1 Example of Dynamic Indecision Producing Oscillations 132
6.2 New Rules Preventing Dynamic Indecision 134
6.3 New Rules Navigating Around a 2m Long Wall 135
6.4 Test Illustrating Sharp Changes in Demand Steering Angle 136
6.5 An Example of Dynamic Windowing 137
6.6 A Second Example of Dynamic Windowing 138
6.7 Test Illustrating Smoother Transitions With Dynamic Windowing 139
6.8 The Effect of the Spreading Constant on Obstacle Avoidance 141
6.9 The Effect of the Spreading Constant on Obstacle Avoidance with Dynamic Windowing 142

6.10 The Effect of Variable Spreading and Dynamic Windowing on Obstacle Avoidance 143
6.11 Lines Used to Generate Avoidance Set Areas. 144
6.12 Generation of Avoidance Sets for a Negative Small Steering Angle. 145
6.13 Common Total Inhibition Avoidance Set Areas. 147
6.14 Initial Definition and Definitions of Varying Length. 149
6.15 The Effect of Total Set Length on Avoidance Performance. 150
6.16 Minimal Avoidance Set Definitions. 151
6.17 The Performance of Minimal Sets. 151
6.18 Avoidance Set Definitions of Varying Widths. 153
6.19 The Effect of Varying Set Widths. 154
6.20 The Effect of Varying the Inhibition Values. 156
6.21 The Final Avoidance Set Definitions. 157
6.22 Results for Avoidance of a 1m Wide Wall. 159
6.23 Results for Avoidance of a 2m Wide Wall. 160
6.24 Results for Avoidance of a 1m Wide Wall Approaching From the Side. . . . 160
6.25 Results for Avoidance of an Asymmetric Wall. 161
6.26 Results for a Narrow Gap Between Poles. 161

7.1 Vehicle Navigating Through a Jumble of Objects. 163
7.2 More Paths Through a Jumble of Objects. 164
7.3 AGV Negotiating a Tight Corner in a Corridor. 164
7.4 The Effect of Sample Time on Avoidance Performance. 165
7.5 Vehicle Faced with a Wide (4m) Wall. 167
7.6 Vehicle Faced with a Narrow (1m) Gap, Goal Aligned With Gap. 167
7.7 Vehicle Faced with a Narrow (1m) Gap, Goal Offset From Gap. 168
7.8 Vehicle Reaching a Dead End. 169
7.9 Sub-goal Navigation Around Wide Walls. 171
7.10 Vehicle Faced with a Narrow (1m) Gap, Goal Aligned With Gap, Sub-goal in Gap. 171
7.11 Vehicle Faced with a Narrow (1m) Gap, Goal Offset From Gap, Sub-goal in Gap. 172
7.12 Vehicle Avoiding a Dead End Due to a Sub-goal. 173

7.13 A Circuit of the Laboratory Avoiding Boxes. 174
7.14 A Second Circuit of the Laboratory Avoiding Boxes. 175
7.15 Negotiating a Corridor Like Object in the Laboratory. 175

A.1 Three Common Shapes for Fuzzy Sets. 192
A.2 Two Overlapping Fuzzy Sets 193
A.3 The Two Different Correlation Methods. 195
A.4 Defuzzification of the Output 196
A.5 Block Diagram of a Simple Fuzzy Logic Controller. 197

B.1 Definition of x, y and θ for the AGV 199
B.2 Angles Used for Triangulation Equations 200
B.3 Calculation of Heading from Position and Beacon Angle 201
B.4 Points Found When Triangulation Angles Are Inaccurate 202
B.5 Illustration of Correction Made for Scanner Position. 204
B.6 Infra Red Detection Circuit, One of Five. 207
B.7 Generation of Reference Level and 'Lost' signal. 208
B.8 Generation of Control Signals. 210
B.9 Logic to Control the Tracking of the Scanners 211
B.10 Illustration of the Field of View of a Scanner. 218
B.11 Set Definitions used in Beacon Assignment. 218

D.1 Main Stepper Driver Isolation Circuit 229

List of Tables

4.1 Linguistic Names for the Goal and Heading Error Angle Sets 64
4.2 Coarse Controller FAM Bank for a Large Distance to the Goal 71
4.3 Coarse Controller FAM Bank for a Medium Distance to the Goal 71
4.4 Coarse Controller FAM Bank for a Small Distance to the Goal 71
4.5 Coarse Controller FAM Bank for a Zero Distance to the Goal 71
4.6 Fine Controller FAM Bank for a Large Distance to the Goal 72
4.7 Fine Controller FAM Bank for a Medium Distance to the Goal 72
4.8 Fine Controller FAM Bank for a Small Distance to the Goal 72
4.9 Fine Controller FAM Bank for a Zero Distance to the Goal 72
4.10 Final Controller FAM Bank for a Large Distance to the Goal 73
4.11 Final Controller FAM Bank for a Medium Distance to the Goal 73
4.12 Final Controller FAM Bank for a Small Distance to the Goal 73
4.13 Final Controller FAM Bank for a Zero Distance to the Goal 73

B.1 Comparison of Triangulation Error Reduction Methods 203
B.2 Results of the Static Tests of Location System 205
B.3 The Scanner EPLD Registers. 213
B.4 Meanings of Status Byte Values 216

C.1 The Root Transputer EPLD Registers. 224
C.2 The Event Timer Control Register. 225
C.3 The Event Flag Register. 225
C.4 The Motion Control Register. 225
C.5 The Scanner Control Register. 226
C.6 The Scanner Status Register. 226

List of Abbreviations

AGV	Automated Guided Vehicle
EPLD	Eraseable Programmable Logic Device
FAM	Fuzzy Associative Memory

Chapter 1

Introduction and Review of Relevant Research

An 'Automated, Guided Vehicle' (AGV) is a vehicle which can guide itself from its current location to some destination within its environment without the aid of a driver or operator. Such vehicles are sometimes described as 'mobile robots' or 'autonomous vehicles'. These terms are often used interchangeably but in general an AGV operates in a structured environment such as a warehouse or factory and is designed to carry out a specific task or tasks while mobile robots are often designed to exhibit 'intelligent behaviours' to demonstrate the application of particular techniques.

The boundary between these two types of vehicle is becoming increasingly blurred as task performing vehicles become more intelligent and intelligent vehicles become more useful. The vehicle which is described in this thesis was developed with the intention of demonstrating how a vehicle, which might be used to carry a payload in a factory, could benefit from the use of more intelligent and flexible control techniques. The techniques used are based upon the fuzzy set theory first expounded by Zadeh [1] and described in Appendix A. They are used to provide a fuzzy rule based controller capable of navigating the vehicle to a desired position and orientation while avoiding unexpected obstacles in its path.

The early sections of this chapter provide a general introduction to the systems which make up an AGV controller and describe some of the different approaches which have been

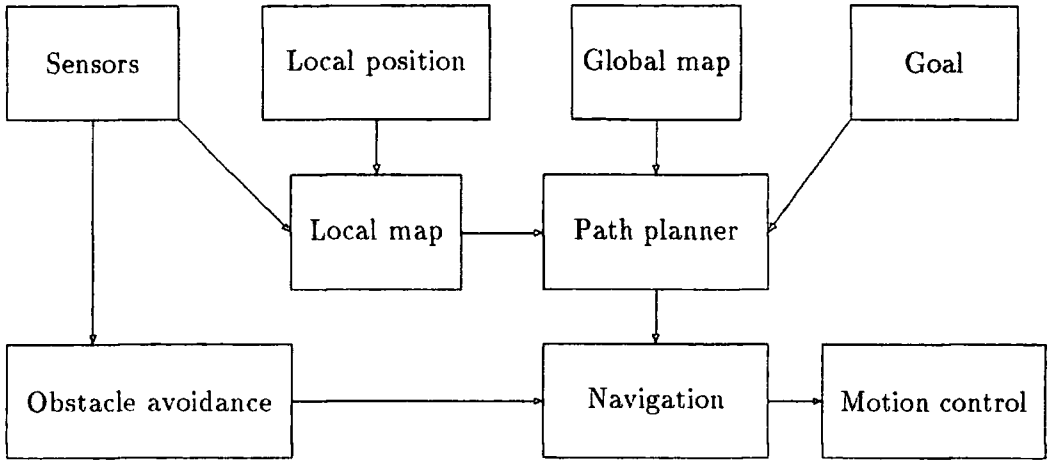

Figure 1.1: Schematic Diagram of the Elements of a Vehicle Guidance System.

used in such systems. Section 1.9 introduces the research papers and ideas which inspired the work described in this thesis to place it in context with previous work. An outline of the systems which operate the vehicle is given in Section 1.10 and finally the structure of the remaining chapters of the thesis, which describe these systems, is presented in Section 1.11 as a guide to the reader.

1.1 Vehicle Guidance Systems

There are many elements which must come together before a vehicle can navigate successfully around its environment. These elements need to be able to cope with a changing environment and enable the vehicle to carry out its assigned task in an efficient manner.

In order to navigate from a start position to a goal an AGV needs to have sensors to gather information on its environment and use this information either to produce an immediate reaction or in some form of path planning. Typically sensor information is used to locate the vehicle and make a local map of its surroundings. The control system may also have access to a global map of its position. This information can be used to plan a path to the goal and information about the path can be passed to the navigation system. Sensor or mapping information may also be used by an obstacle avoidance system to influence

the navigation controller. A low level motion control system may be used to translate navigation demands into actions. A schematic of some of the possible relationships between these elements is shown in Figure 1.1.

AGV guidance systems may have some or all of these elements depending on their purpose and the approach favoured by their designers. The design of individual elements and how they can interact is discussed in more detail in the following sections which give a brief introduction to the some of the important elements of automated vehicle design.

1.2 Sensor Systems

Sensing is vital for the control of automated vehicles. Without sensor information it becomes impossible to find out where the vehicle is and plan a route to its goal. Sensors provide the information needed to locate the vehicle, map its surroundings and avoid collisions. There is a wide variety of sensors used in robotics the most common of which are described here.

1.2.1 Ultrasonic Transducers

The time taken for a sound wave to travel to an object and be reflected back provides a useful measure of the distance to an object. Ultrasonic transducers have been widely used in robotics applications, the Polaroid [2] transducers being very popular due to their long range. Ultrasound sensors give good range measurements but have a wide beam angle and so have bad angular resolution, making it difficult to find doorways and small openings. They also suffer from the change in the speed of sound with air temperature [3] although with suitable calibration [4] this can be allowed for. Ultrasound sensors are relatively cheap and generally accurate but also suffer from specular reflection from nearby objects and deflection by non-perpendicular targets. The frequency of measurement is limited by the speed of sound.

Attempts have been made to allow for these problems. Kuc [5] derives a mathematical model of the beam and scans the sensor so enough data is available to guarantee a clear area ahead. Everett [3] suggests overlapping sensor beams to produce better angular resolution. Flynn [6] combines ultrasound and infra red data to locate doorways in an environment

while Steer and Atherton [7] suggest the use of 'acoustic cat's eyes' to improve the sensor's performance.

Ultrasonic transducers are usually mounted in one of three ways. In a fixed ring to give a 360 degree scan, on a rotating scanner or fixed facing in a particular direction of interest. Rings and scanners are generally used in association with mapping techniques while fixed sensors are usually for obstacle avoidance. A considerable amount of work has been done by Barshan and Kuc on the characterisation and use of sonar data [5] [8] [9] [10].

1.2.2 Sonar Arrays

In place of a single fixed ultrasound sensor or a scanning sensor an array of many transmitting and receiving elements can be used. An array is capable of gathering more information than is possible from one ultrasound transducer. Most arrays are currently in the development stage. Vietz [11] describes an array which produces an electronically steerable beam via interference effects which can produce scanning with no moving parts. Munro et al [12] describe an array which it is hoped will produce a uniform beam from which angular information over a wide beam angle can be derived by a receiver.

1.2.3 Infra Red Sensors

Infra red sensors provide good angular information but are not very accurate in finding range information. Hu [13] has produced a ranging sensor by using a laser diode and measuring the amplitude of the return signal. These sensors suffer badly from drift and require an ultrasonic transducer for calibration. Being light based the frequency at which readings can be taken is very high and the good angular resolution of these sensors makes them very attractive.

1.2.4 Lasers

Accurate, range measuring, laser systems are very expensive but produce a lot of very accurate range and angular information. Frolich et al [14] describe a ranging camera which produces a depth picture of the area in front of their robot very quickly. Cox [15] uses an optical ranger to give a high speed scan of the surroundings. A laser system has a very

high level of performance but this must be weighed against the high cost and complexity. A large number of simpler sensors can be used for the cost of a laser system.

1.2.5 Vision

Vision provides a considerable quantity of information which requires heavy processing. Studies of vision navigation have been consulted only in passing as there are no plans to implement vision systems in Durham at present due to the high cost which makes them unattractive for warehousing applications.

1.2.6 Odometry

Almost all AGV guidance systems include odometry to keep track of their movements. This consists of monitoring the rotation of the vehicle's wheels using shaft encoders to discover how far the vehicle has moved. As has been pointed out by Pears [16] odometry suffers from errors due to wheel slip and measurement uncertainties but the feedback provided is vital for ensuring correct motion control.

1.2.7 Other Sensors

Many other sensors have been used or proposed. Bradshaw [17] has reviewed the sensors available and has made predictions on the possibilities of new sensors, such as millimeter radar, becoming widely available. Vehicles are often fitted with local sensors such as proximity detectors and limit switched bumpers to give warnings of close objects. Large robot vehicles operating outdoors may use navigation satellites and compasses. Short range local sensors are very useful for detecting objects missed by ranged sensor systems and can provide a final safety barrier.

1.3 Localisation

Knowing the location of a vehicle is vital if any map is to be followed and start and goal positions are to have any meaning. Most AGVs locate themselves by knowing their starting

locations and using odometry to keep track of their location thereafter. The problem with this is that large odometric errors can build up. Many workers have opted to ignore these errors and concentrate on other problems. Elfes [18] noted the blurring effect that including the position errors had on his sensor derived maps, ultimately making their information useless. Several methods have been used to overcome this. HILARE uses active infra red beacons to triangulate position and the GEC Caterpillar and Oxford Turtle [13] use location bar codes. Soldo [19] keeps track of the motion of vertical edges relative to his robot to correct errors while Pears [20] used wooden correction boards. Recently work has been published on using 'geometric beacons', that is static environment features, recognised by optical [15] or ultrasonic [21] scanning. Tarrassenko et al [22] propose to use neural networks to recognise the range measurements from many positions to place the vehicle accurately. Kalman filter algorithms are commonly used to get the best estimate from a number of sensor sources.

1.4 Mapping

Maps provide a way of storing information about the environment that an AGV is to operate in and can be used when there is no current sensor information available for a particular area. If the exact global location of the vehicle is known then a global map can be built up. Otherwise information is strictly local.

Much work on mapping has been concerned with how to acquire the information from sensors [6] [9]. The validity of the maps is proven by comparison with a manually acquired map. Some workers [23] [19] have argued that mapping adds an additional and unnecessary layer between sensing and action. Maps they believe are an abstract representation of the real world which aid human but not machine understanding.

Nevertheless much work on path planning which has been done in simulation assumes the existence of a map or large amounts of accurate sensor data. The continual availability of up to date sensor information may be a reasonable assumption with high frequency systems such as the laser used by Hoppen et al [24] but not with slower more inaccurate sensors such as ultrasound.

The simplest mapping format is a grid with areas marked as full or empty. This method

was used in all early robot mapping schemes and is still used where researchers believe their sensor data is reliable enough to make this assumption hold [9].

The most interesting technique to date is that of certainty grid mapping [18]. In this technique an area is divided into a grid of squares, each of which has an 'occupancy value' between -1 and 1 representing the probability that an obstacle is present. A value of zero represents unknown while negative values represent the degree of certainty that the area is clear and positive values the degree of certainty that it is occupied. The occupancy probability is found from sensor readings and such a method is well suited to sonar sensors which tend to give spurious readings and have a wide beam width giving rise to uncertainty as to the exact position of an object.

The certainty grid method has been used and modified by other workers. Choo [25] has produced a more mathematically rigorous method of updating the probability entries in the grid and has experimented in his simulation with different sensor beam widths. From this he has concluded that narrow angle sensors are good for finding small objects, corners and openings while wider angle sensors, such as sonar, spot large clear areas faster.

Borenstein and Koren [26] have taken the opposite approach and use a simplified updating technique known as 'Histogramic in Motion Mapping'. In this method they use the assumption that the the echo response from an ultrasonic transducer is always from a target in the center of the beam pattern at the measured range. This allows them to make fast updates of the map and hence use it to initiate avoidance of unexpected obstacles even with relatively fast moving robots.

An alternative approach to the grid mapping methods is to store information about the environment as a set of nodes and paths in a tree-like data structure. This aids the execution of planning algorithms. Hoppen et al [24] produce a set of primary and secondary nodes which mark clear paths for a robot to follow. Shaout and Isik [27] propose a method of obstacle storage as fuzzy sets to enable matching with inaccurate, incomplete and uncertain sensor data.

The mapping technique used for a particular vehicle depends on both the sensors used and the planning algorithms to be applied from the map. While a grid map is easier to visualise and use for a human it is not necessarily the best approach for a computational solution.

1.5 Path Planning and Reactive Control

Path planning has been discussed in many papers, a large number of which have been purely theoretical. Some automated vehicles still use off-line path planning or rely on human direction to some extent but the complexity of the path planner reduces the amount of information which must be given to the vehicle. The task of choosing a route for a vehicle to follow is very complex and full of pitfalls. The time taken to plan a route must not be excessive and yet the route should not be so badly planned that it takes far longer than necessary. In well constrained environments vehicles can have sets of stored routes they follow similar to following a physical line or wire. More independent vehicles must plan from a map they have been given of their environment or have acquired through sensor data.

Researchers looking for optimal solutions have produced complex routines for path planning in static and dynamic environments but they require almost total knowledge of that environment and make no allowances for inaccurate data. Shih et al [28] propose a matrix representation of all faces of all objects in the environment and their motion and produce a 'near optimal' solution. The computational load of such planning is excessive as is shown by Delamadrid [29] who treats all objects as moving cylinders but found that a real robot moved only a few feet a minute in an environment with only three obstacles.

Other workers have tried to produce a solution which works off limited, local, information [30] or to guarantee a solution even if a long path is generated [31]. The reasoning behind this 'guaranteed convergence' method is that if nothing is known about the environment except the location of the goal it is more important to be sure that the vehicle will reach that goal than how fast it can do it.

Path planning is computationally intensive and can deal only with the information which is known about the environment. In most applications however the information about the environment is incomplete, suffers from inherent inaccuracies and noise in sensor data, and is constantly changing. In this case then, planning a precise path to follow may well be a waste of time as new sensor data and changes in the environment may well ruin a detailed plan.

The alternative to path planning is a reactive control scheme, that is a scheme where robots are given certain reactions to sensor information which combine to give an overall

'behaviour' which will enable the robot to achieve its goal. Brooks [23] argues that by giving robots sets of behaviours to interact with their environment it is possible to produce 'autonomous creatures' who can cope with changes in the environment much better than by relying on plans. This has lead to a highly successful series of robots at MIT [32] using his 'subsumption architecture'.

Another commonly used technique for fast reacting vehicles is the force field approach in which an attractive virtual force [33] [34] attracts the vehicle to its goal and obstacles, paths etc. produce virtual repulsive or attractive forces to guide the vehicle. This method is simple and fast but suffers from the problems of local minima in the virtual force field which may cause the vehicle to become stuck. The vehicle can react quickly to changing events because it has a simple plan or reaction to each perceived event but it has no way of ensuring that the accumulated result of these responses is desirable.

Slack [35] has proposed the combination of virtual force fields with 'navigation templates' which are in effect additional behaviours planned into the map. Therefore obstacles might be attached to navigation templates indicating that the vehicle should steer to the left of a particular obstacle or move in a certain direction. They provide a method of combining guidance from a high level planner with a lower level reactive scheme.

1.6 Local Obstacle Avoidance

The issue of local obstacle avoidance is tied in very closely with sensors and path planning. A path planner which continuously alters its strategy to take account of new sensor-derived information should avoid obstacles itself and need only have an emergency stop function if the planner has obviously failed.

If pregenerated plans are used or the vehicle simply sets off towards its goal it must be able to move around unexpected obstacles. The previously mentioned virtual force field method [33] is a popular and effective method of avoiding obstacles. Borenstein and Koren have used it in conjunction with a human operator to give 'Teleautonomous Guidance' [36] where a low level routine produces obstacle avoidance vectors and a joystick input gives

a vector to travel along. Payton et al [37] point out that an obstacle avoidance routine must be closely linked to the ultimate aim as independent routines can produce routes detrimental to the overall goal.

The avoidance of obstacles can be part of a carefully conceived plan, or a temporary exception to a globally planned route, or a semi independent behaviour depending on the control architecture. A route planner which continuously updates its plan every time an obstacle is detected is heavy on computation, while simple reactive schemes can send the vehicle in undesirable directions. Most physically implemented schemes are a combination of the two.

1.7 Motion Control

The motion control of mobile robots is difficult to separate from the other layers. In many designs there is no mention of the controller so it must be assumed that conventional control techniques are used to achieve the demanded speed from the motors. AGVs such as those produced by Pears [16] and Cox [15] follow an imaginary line and P.I.D. control keeps them on those lines and controls their return to them if they depart from the line. If a plan is made to give an optimum route between points then the ability of the vehicle to follow that route depends on its controller.

The feedback used to control motion is often provided by changing the demand direction as in virtual force field techniques. In these cases it is assumed that a simple motion controller will be sufficient. By using a more sophisticated intelligent controller it is more likely that the vehicle will perform as desired and if it is adaptive the controller can cope with changes in vehicle dynamics.

There have been some experiments carried out using control by artificial intelligence techniques such as rule based control, fuzzy logic and neural networks. These methods are more than just motion controllers often performing some of the obstacle avoidance and local path planning functions. Of particular interest are the fuzzy methods which are further discussed in the following section.

1.8 Fuzzy Logic

Fuzzy logic offers a powerful tool in the field of control. Its basic premise that all data is only partially correct is an excellent way of handling noisy and unreliable sensor data. It can control systems which conventional controllers find hard to manage and no mathematical model of the system is needed.

Li and Lau [38] have shown a fuzzy logic controller can perform as well as, if not better than, a model reference adaptive controller. Kosko [39] has shown fuzzy logic can perform as well as a Kalman filter algorithm and both found fuzzy controllers to be robust to system parameter changes. In the field of robotics Payton et al [37] when discussing problems occurring with control of an all terrain AGV call for a fine grained architecture where information is known to all subsystems. They say that decisions should be made on the basis of weights for the desirability of various actions being produced by the controller. This is precisely what fuzzy logic does.

Some efforts have been made to use fuzzy control in mobile robots. Maeda et al [40] have used a fuzzy steering algorithm to control a vision guided model robot on a scaled down road. Garcia et al [41] describe the use of a fuzzy controller in a simulation to control a wall following robot. Kosko [39] uses a fuzzy steering control algorithm to back up a simulated truck into a parking space. More recently promising work on fuzzy vehicle guidance has been reported by Yen and Pfluger [42] and Saffiotti et al [43] which is covered in more detail in Section 1.9.1

There is not a well founded theory for the design of control systems using fuzzy logic and in most cases the rules used are designed using a combination of designer knowledge and trial and error. Several possible methods have been proposed for including adaptive elements to design and modify a fuzzy controller. One such method is that proposed by Zhou and Qui [44] which uses an expert system to select various fuzzy rule sets which are tuned on-line according to a rule based tuning mechanism. Other approaches are to use neural networks [39] or genetic algorithms [45] to learn appropriate rules and set definitions. Even without adaption fuzzy logic provides a powerful method for applying operator knowledge to control systems which are hard to analyse mathematically.

1.9 Discussion of Relevant Literature

The preceding sections have given an introduction to the many elements which must be combined to build a successful automated vehicle and have described some of the approaches used by previous researchers. More detailed information about early AGV projects can be found in the reviews of Pears [16] and Hu [13].

The work in this thesis concentrates on the areas of *obstacle avoidance* and *navigation.* The other elements have either been replaced by pre-programming, as in the case of path planning and mapping, or by simple systems not involving the use of advanced techniques, as is the case with the localisation system. The control has been based on fuzzy logic techniques which are described in Appendix A.

This section introduces previous work which was particularly influential in the selection of the area of research for this thesis and in guiding some of the decisions taken during the course of the research. Leading on from this Section1.10 outlines the research undertaken and briefly introduces the new techniques involved.

1.9.1 Planning versus Reaction

Approaches to the control of automated vehicles and mobile robots can be divided into two main types. They can be characterised as the planning approach which is a 'top down' method and the reactive approach which is a 'bottom up' method. The planning methods involve converting all the available information into a form which can be acted on by an intelligent planner which then decides upon the path the vehicle should take. In the simplest case these paths are predetermined tracks marked by white lines or magnetic strips which the robot follows. More sophisticated methods involve following line or curve segments generated by the planner such as is used by Cart Blanche [15] and the previous work at Durham [16]. The advantage of planning is that it enables a high degree of control over the path taken and can cope with maze like structures where the path is not obvious. The main problem is that the generation of plans requires a large amount of computation and the result is often inflexible and unable to cope with an uncertain or changing environment.

The reactive approach is guided mainly by ideas from studies of animal behaviour and Artificial Intelligence research. The idea is to use a close linking of sensors and actions

to enable vehicles to survive in a real environment which changes and is detected using noisy sensors. This provides robots with low level intelligence but makes it harder to ensure they complete assigned tasks. One reactive method is the 'Subsumption Architecture' championed by Brooks [23] to develop 'task achieving behaviours' in small robots which are regarded as being 'autonomous creatures'. The subsumption architecture consists of a series of finite state machines (4-bit microprocessors) which each have access to a set of sensors and the motor output. Each finite state machine reacts in a certain way to sensor input and its output will affect the robot motion unless it is overridden by a state machine higher in the control hierarchy. Arkin [34] also uses biological parallels to support his *motor schema* philosophy. This combines sensor input and motor control in individual behaviours which all contribute towards an overall response. The vector outputs of individual behaviours are summed and give the final control output which is based on various biological studies into obstacle avoidance strategies in animals. Basic schemas include 'Stay on path', 'Avoid static obstacle' and 'Move to goal'. Arkin's approach, the combination of many low level behaviours, is similar to that of Brooks [23] but he sums behaviours rather than having low level behaviours overridden by higher level ones.

The more conventional planning approach gives robots the ability to be useful in industrial or domestic situations for which they are specifically designed. The 'autonomous creatures' approach gives more flexible vehicles, better able to cope in the real world but harder to predict. Ultimately the two approaches will converge and this is indeed happening as shown by Arkin's ventures into industrial applications [46] and the combination of ideas in projects such as Oxford's Turtle [13].

Of particular interest is the paper 'Plan Guided Reaction' by Payton et al [37] which discusses some of the problems encountered during the ALV, Autonomous Land Vehicle, project. In this paper the authors discuss how their robot operating from a series of plans reacted unexpectedly in the real world. They also note that one of the major problems was that separately designed and implemented techniques for navigation and obstacle avoidance produced problems because the functions are not in reality independent. They therefore proposed a 'fine grain' approach to the design of AGV control systems in which different subsystems (such as navigation and obstacle avoidance) would produce several suggestions as to the desired control output. The final output would then be based on a comparison of these suggestions to produce the most suitable result for differing goals.

This type of procedure is very closely analogous to a fuzzy logic system where inputs activate the output rules to varying degrees and the final result is based on a combination of all these rules. This similarity inspired the work described in this thesis in which the AGV control system is based upon fuzzy logic techniques and combines both the obstacle avoidance function and the navigation function. The two functions are combined prior to the defuzzification stage of the controller and produce a final, defuzzified, control output which satisfies both the aim of moving towards the goal location and of avoiding collisions.

Some recent work along similar lines has been undertaken by Yen and Pfluger [42] and Saffioti et al [47] [43]. Yen and Pfluger consider a cylindrical robot guided to its goal by a set of simple fuzzy rules. They identify the problem with using inhibitive rules, namely the unsuitability of standard defuzzification methods and propose a possible alternative technique. Saffioti describes the work undertaken on the robot SHAKEY which examined the context weighted combination of many simple fuzzy rulesets for different tasks with promising results.

1.9.2 Mapping

As has been described earlier in this chapter a particularly interesting mapping technique know as the *certainty grid* method has been applied by Elfes [18]. This method divides the area around the vehicle up into a grid and assigns an occupancy value to each square on the grid to indicate the probability that there is an object in that grid square. This method has several valuable properties. Firstly it can fuse the data from several sensors to give a more accurate picture of the surrounding area. Secondly the information held in the grid can be used by routines for navigation and obstacle avoidance without needing to know how the information was acquired, thus improving portability. Thirdly the information being of a probabilistic nature would seem to be ideal for processing by a form of probabilistic or fuzzy logic. Although some work has been done on sensors the research vehicle does not, at present, have the capability to detect obstacles. Therefore a predetermined global occupancy grid is used consisting of ten centimeter squares each of which has an occupancy value of between one and 100 associated with it representing the percentage probability of an obstacle occupying that grid square. This is described more fully in Section 5.2.

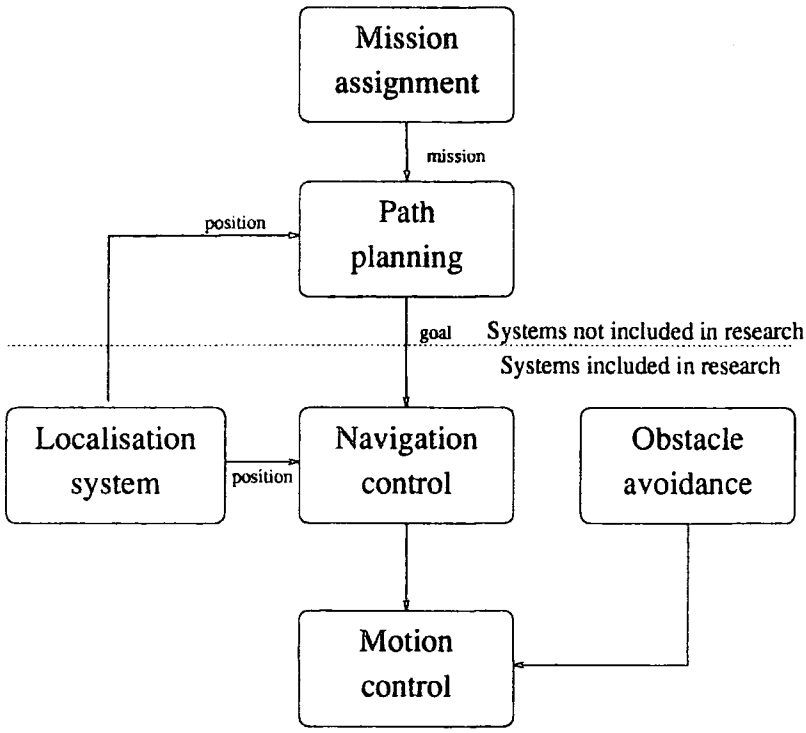

Figure 1.2: Layers in the Control Hierarchy

1.9.3 Control Hierarchy

In almost all complete mobile robot systems there is a control hierarchy. As Slack [35] describes this usually takes the form of higher level functions coping with planning and reasoning problems such as mission planning and long term navigation while lower levels are most intimately concerned with the robot's sensors and short term problems such as local obstacle avoidance. Generally speaking the higher a task is in the hierarchy the more intelligence is required but the less attention it pays to immediate problems.

For large groups of researchers involved in developing a vehicle it is possible to consider all levels of the hierarchy and to design and test advanced systems for all the various subsystems. For smaller groups however it is necessary to concentrate on certain elements and to simulate other layers or replace them with a human operator. Figure 1.2 shows the hierarchical control scheme assumed to be behind the control of the vehicle used at Durham. The mission assignment level could be dealing with many vehicles in a warehouse or just one but it would assign missions perhaps involving the moving of goods from storage ready to be shipped from the warehouse. The path planning stage would use a global map to assign goals to the vehicle such as 'go to loading bay 6' and pass this goal to the navigation controller in the form of target coordinates and a target orientation. These layers have

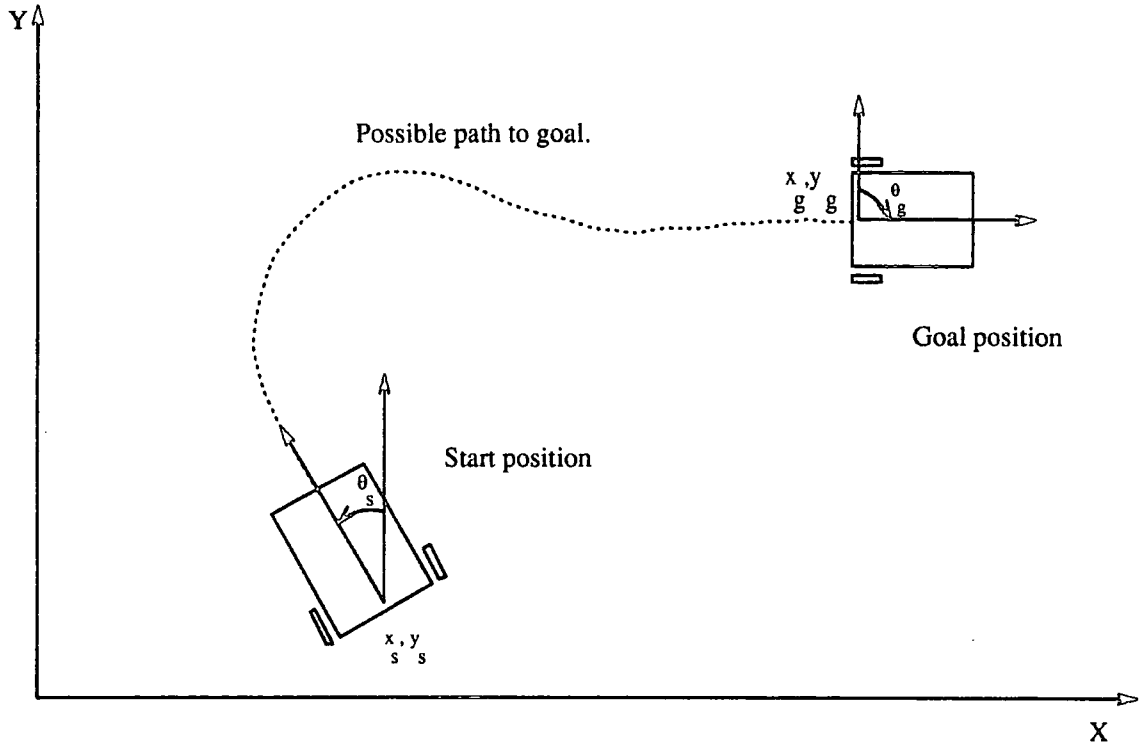

Figure 1.3: The Navigation Problem

not been implemented and the navigation controller is therefore passed preset goals by the operator.

The lower layers which have been implemented are those necessary to guide the vehicle from one goal position and orientation to another without colliding with obstacles in the path of the vehicle. How this has been achieved is outlined in the following section.

1.10 Outline of Research

The navigation problem which this research has been focused on is that shown in Figure 1.3 where a vehicle is at position x_s, y_s and orientation θ_s and is required to move to x_g, y_g and end with orientation θ_g. For a vehicle able to turn without moving the obvious solution is to simply move to the goal location and rotate until the required orientation has been reached. This option is not available for conventionally steered vehicles which makes the control more complex and the controller must include a method for avoiding obstacles.

The control method for the research vehicle consists of four main parts, a localisation system, a navigation controller, an obstacle avoidance system and a motion controller. The

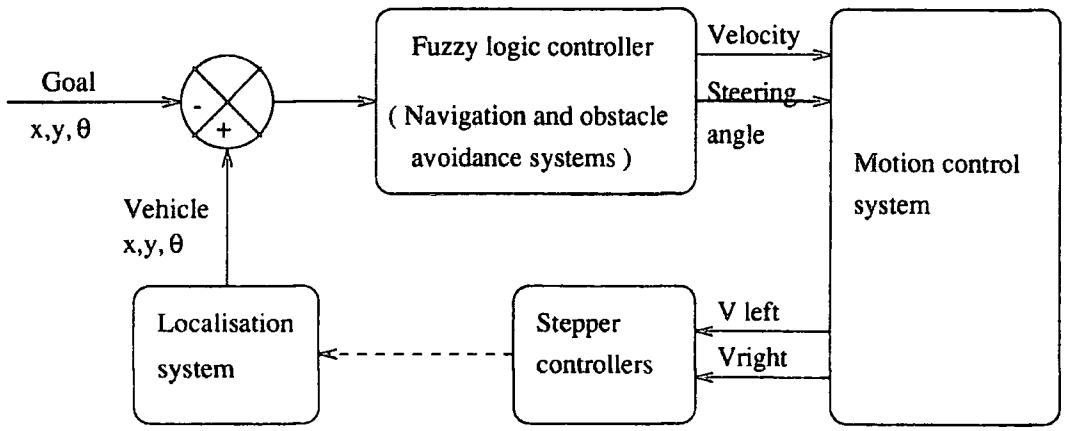

Figure 1.4: Basic Control Loop for the Research Vehicle.

control loop which is operating on the vehicle is shown in Figure 1.4. The localisation system provides the controller with an up to date estimate of the vehicle's current position based upon sensor information. The errors between the current position and orientation and that of the goal are used by the navigation controller in conjunction with the obstacle avoidance system to produce demand values for the velocity of the vehicle and the steering angle which needs to be applied. These demands are then passed on to the motion control system which translates the demands into the required velocities of the drive wheels, applying some limits to the acceleration and rate of change of steering angle to keep within the dynamic capabilities of the vehicle.

The localisation system consists of odometry carried out by shaft encoders fitted to the driving wheels of the vehicle and a beacon location system. The beacon location system uses infra red detecting scanners on top of the vehicle to locate fixed beacons transmitting a continuous infra red signal and triangulates the vehicle's position by measuring the angles to the beacons. This provides a simple and cheap method of correcting the errors in the odometric estimate of the vehicle's position.

The navigation controller applies a set of fuzzy rules to the errors between the current and goal position and orientation. This produces a *fuzzy fit vector* which contains values indicating the desirability of each of seven output sets. These output sets range from turning sharp left to sharp right and in a standard fuzzy controller would be defuzzified straight away, but instead the control method combines the fuzzy fit vector with a *mask vector* produced by the obstacle avoidance system.

The obstacle avoidance system works by examining areas of the occupancy grid close

to the vehicle. These areas are know as *avoidance set areas* and are linked to the output sets. The area linked to the hard left steering set, for example, covers the area which the vehicle would turn into if it steered hard left. The obstacle avoidance rules are *inhibitive* rules and produce a *mask vector* which contains a value for each possible output steering set indicating how desirable it is to steer in that direction. These values range from zero, indicating collision with an object is certain, to one, indicating an area free from objects in that direction.

The two vectors are combined by using a number of new techniques to supply a resulting vector which can be defuzzified to give the desired steering angle. A technique known as *rule spreading* is applied to the navigation fit vector to ensure all possible steering angles have some activation. The vectors are combined by multiplying the value in the fit vector by the corresponding value in the mask vector. A technique known as *windowing* is then used to mask out values in the resulting *combined fit vector* not grouped around the maximum value. This is to prevent the defuzzification stage from producing an output identified as being undesirable by the obstacle avoidance controller which is possible with conventional defuzzification techniques.

After defuzzification the resulting demand steering angle is passed to the motion controller along with a demand velocity produced by a simple fuzzy logic velocity controller. The motion control system converts this demand steering angle and velocity into stepper motor speeds to drive the vehicle. The controller has limits on the rate at which it can achieve the demanded velocity and steering angle which represent the vehicle dynamics, in this case the stall characteristic of the stepper motors.

The main original elements of the research lie in the techniques used to combine the output of the navigation controller and obstacle avoidance controller. Also of importance is the way in which the geometric shape of the vehicle and its steering mechanism is considered in order to produce the obstacle avoidance sets. Most work on mobile robots uses cylindrical vehicles able to rotate without moving, not the more conventional shape and steering configuration used here.

1.11 Thesis Outline

This chapter has given an introduction to the field of automated guided vehicles and fuzzy logic control. The work of some other researchers has been reviewed and placed into context with the research covered in the rest of this thesis. The next chapter covers the vehicle itself, its drive and steering mechanism, physical subsystems and onboard computer system. The software used for direct control of the vehicle is also described and the way control is distributed amongst a large number of inter-related parallel processes explained. Software which simulates the motion of the vehicle is described in Chapter 3 along with the software developed to realise fuzzy logic controllers. This chapter need not be read in sequence with the others, it simply provides more information about the simulation and control software used to develop and implement the methods described in this thesis.

Chapter 4 describes the development of a fuzzy logic based navigation controller to drive a conventionally steered vehicle from an initial position and orientation to a goal position and orientation. The improvements achieved during the design process are illustrated by a number of tests. The set definitions and rules used are presented. The issue of velocity control is also examined and another fuzzy logic controller presented which controls the velocity of the vehicle.

The next chapter deals with the obstacle avoidance function. The reasons behind the adoption of an inhibitive ruleset with set definitions linked to areas of an occupancy grid are discussed. New techniques, *windowing* and *rule spreading*, developed for integrating obstacle avoidance with the navigation controller are introduced. The way the obstacle avoidance function works and how it is related to the navigation controller is explained both in theory and by the use of a series of examples showing the outputs from the controller at several stages during a test.

The performance of the control system in a series of tests is presented in Chapter 6. The results are discussed and used to demonstrate why some modifications to the control scheme were necessary. The technique for generating obstacle avoidance sets from the vehicle geometry is described and results from some tests which lead to the selection of the final avoidance set definitions are given. Finally results from the research vehicle using the settings arrived at from the experiments presented earlier in the chapter are shown. They demonstrate the vehicle avoiding a variety of obstacles.

More complicated tests involving larger and more densely packed obstacles are presented in Chapter 7 along with some situations where the basic low level control method is unable to cope. The way in which higher levels of the control hierarchy might be able to cope with such situations by the introduction of sub-goals is discussed. Results from a number of runs undertaken by the research vehicle are then shown, illustrating the more complex tasks achievable using sub goals.

The final chapter presents the conclusions which can be drawn from this research and considers what further work might be undertaken leading on from it.

Chapter 2

AGV Hardware and Associated Sofware

In order to carry out research into the performance of a vehicle control system it is vital to have a vehicle which can be used as a testbed. While it is possible to carry out some design using simulation, the effects of noise and uncertainty can only be seen when a controller is used in practice. The vehicle used for this research is a modified version of that used by Pears [16]. The vehicle is approximately 60 centimeters wide, 80 centimeters long and 80 centimeters high. It has three wheels, the two rear ones being powered by stepper motors and their rotation monitored by shaft encoders. Three infra red scanning sensors are mounted on top of the vehicle as part of the location system. The vehicle is controlled by a computer system consisting of two transputers and some dedicated interface electronics. The transputers can be connected to a personal computer for program development and exchange of data about tests. The software which runs on the transputers was written in 'C' and is arranged as a number of communicating parallel processes. Each process has its own task, such as driving the stepper motors, keeping track of the vehicle's position or applying the fuzzy rules.

This chapter first describes the vehicle, its onboard electronics and computer system. Then the overall structure of the parallel processes running the various control systems and subsystems is presented. A brief description is given of the tasks performed by each process. The simple systems of motion control and localisation are then described. An outline is

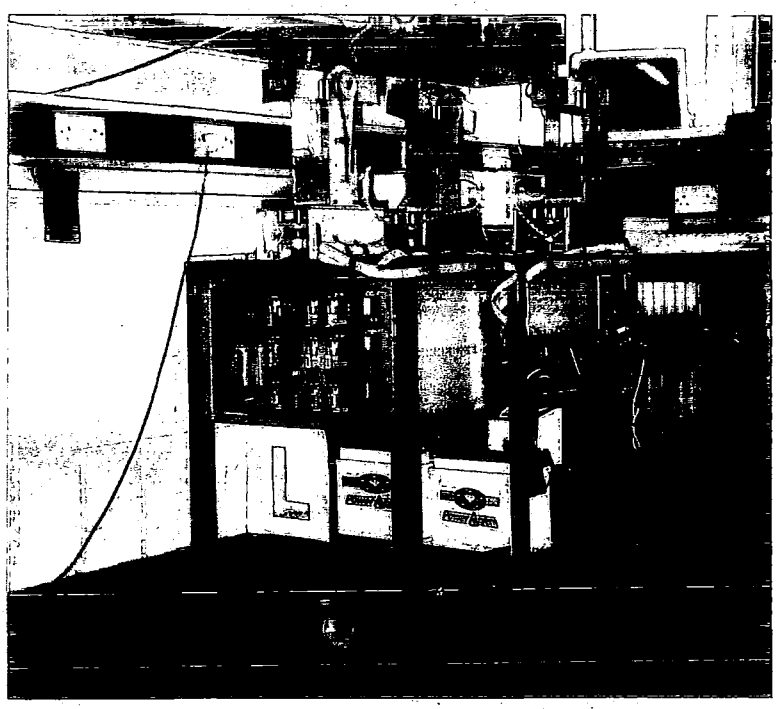

Plate 2.1 The Research Vehicle

given of each system and the way in which the tasks are perfomed by the use of software processes, interface electronics and sensors is described. Some of the low level software used to control the motion of the vehicle is described in more detail. This includes derivation of the equations of motion and the restrictions applied by the motion controller software to allow for the dynamics of the vehicle and its motors.

2.1 The Vehicle Hardware

2.1.1 The Vehicle

The vehicle is shown in Plate 2.1 and a plan and side elevation of the vehicle can be seen in Figures 2.1 and 2.2 respectively. The rear two wheels are powered while the front wheel is a swivelling castor wheel. Steering is provided by controlling the relative velocities of the two rear wheels. The rear wheels are driven by Superior Electric 'Slo Syn' 3V, 4A stepper motors geared down in the ratio 7:36 by the use of timing belts. The drive for the stepper

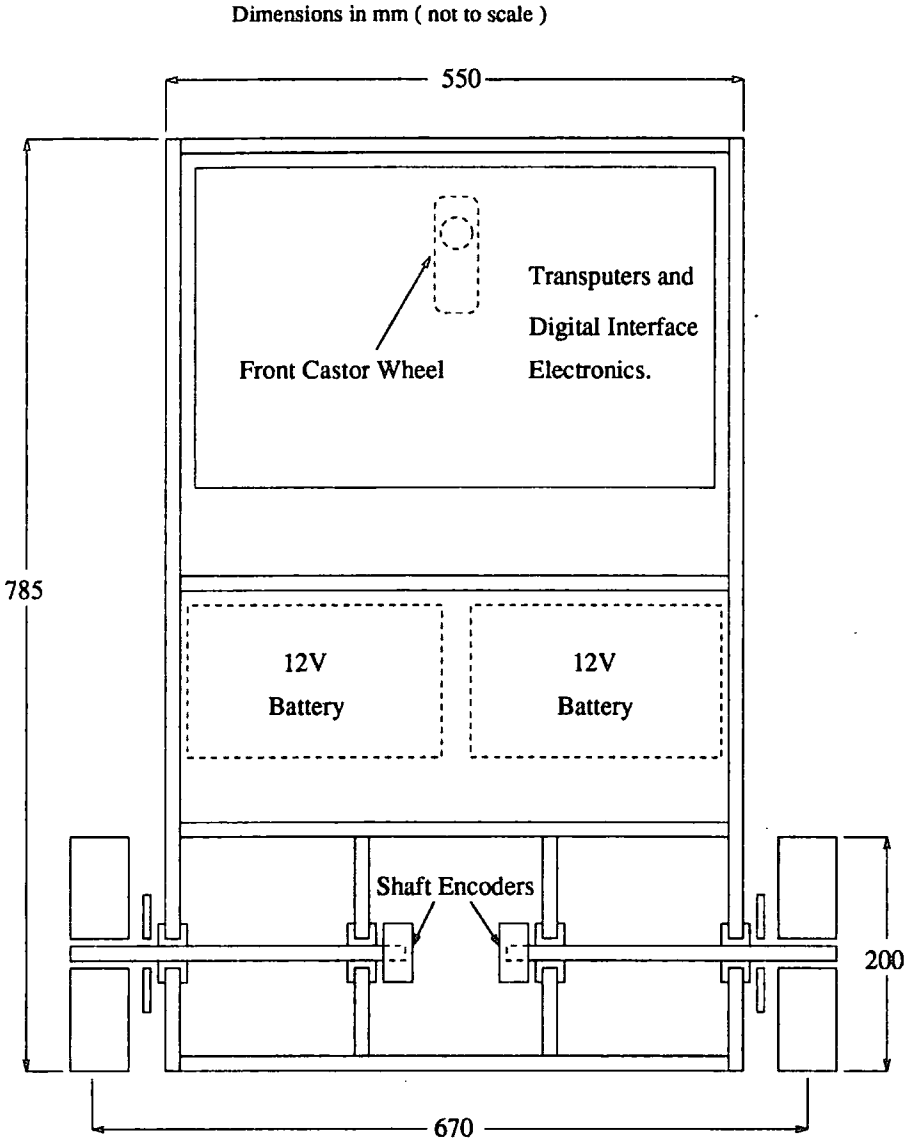

Figure 2.1: Plan View of the Research Vehicle

motors is arranged so that it can be switched between low and high current. This allows the battery power to be conserved for normal operation but allows additional torque to be available if it is desired to be able to perform tight turns at higher speeds.

The rotation of each of the rear wheels is monitored by a HEDS 5640 A06 shaft encoder with a resolution of 500 pulses per rotation. This gives information on the speed of the vehicle and gives a mechanism for detecting a stall by the motors. Three infra red receiving scanners are mounted on top of the vehicle and form part of the location system of the vehicle which is described in brief in Section 2.6.2 and in more detail in Appendix B

Analogue interface electronics for the stepper drivers, location system and odometry are

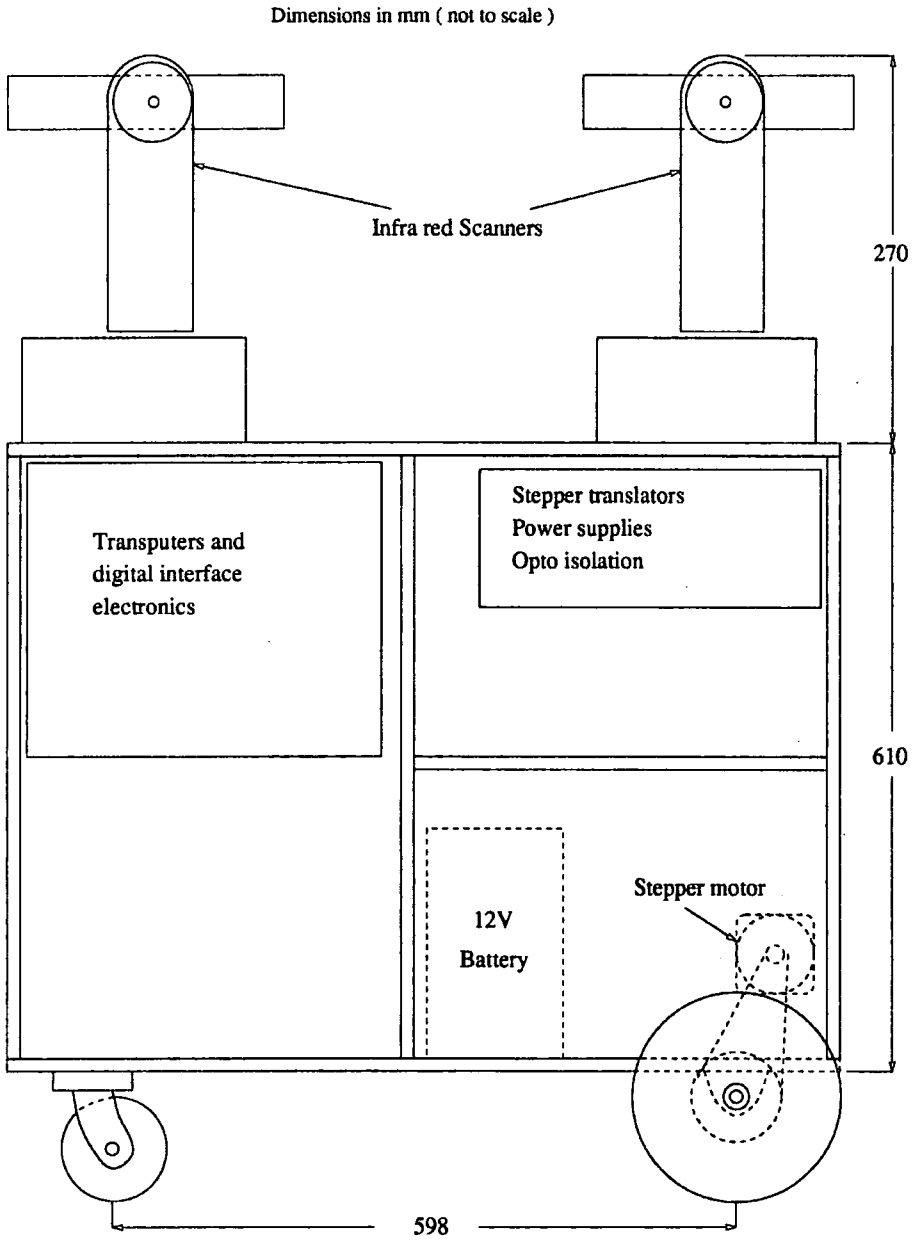

Figure 2.2: Side View of the Research Vehicle

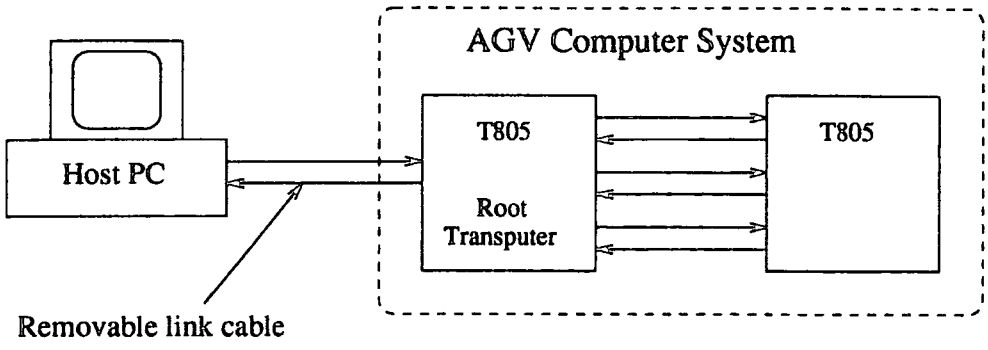

Figure 2.3: Host PC and Onboard Computer

mounted at the back of the vehicle in a '3u' sized rack. The transputer based computer system and EPLD based digital interface electronics are mounted in a '6u' rack at the front of the vehicle. Both of the stepper motors and all onboard electronics apart from the computer system are powered from two 12V car batteries mounted centrally on the vehicle.

2.1.2 The Computer System

The computer system consists of two transputer boards and is shown in Figure 2.3. The root transputer, which is connected directly to the host PC, has 16M of memory which allows data from long test runs to be stored and downloaded after the end of the run. This transputer runs most of the processes concerned with the low level control of the vehicle. The second transputer has 4M of memory and runs the higher level control functions, that is the fuzzy logic navigation system and the obstacle avoidance system. The two transputers are linked by their own high speed communication links and connected via a slower link channel to a PC for software development. The PC is itself linked to the departmental network enabling data from test runs to be processed and displayed using the graphics utilities available on workstations. While it is performing a run the vehicle operates independently from the host PC but it is reconnected after the run is complete and information about the vehicle's performance is transferred to the PC.

Transputers were chosen because they provide a good environment for running a real time control system. A large number of independent processes can run on each processor performing different tasks such as controlling the driving wheels, tracking the vehicle's position and generating control commands. The T805 processor boards used were supplied by the Durham University Microprocessor Centre who were therefore able to help with the problems of interfacing between the transputers, the PC and external sensors.

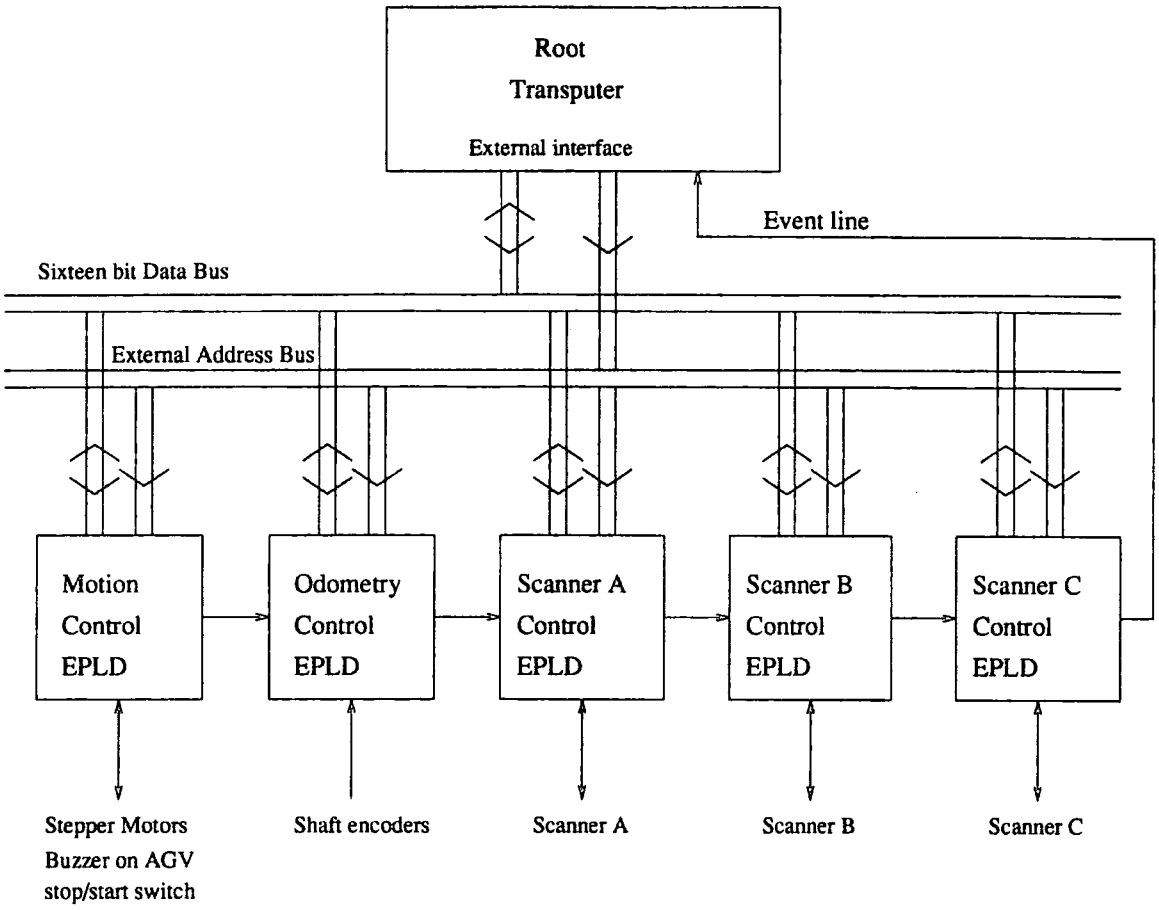

Figure 2.4: Subsystems Realised in Digital Logic

The programs to control the vehicle were written using the INMOS Parallel 'C' toolset which is an expansion of the ANSI 'C' standard capable of utilising the parallel processing power of transputers. 'C' was used to maintain compatibility with the previous software for the vehicle and to enable routines to be transferred directly from the simulation on the PC to the vehicle. This would not have been possible using the transputer specific language OCCAM.

2.1.3 Digital Interface Electronics

A number of digital subsystems, shown in Figure 2.4, are connected to the computer to perform low level control of the sensor systems. The subsystems interface between the computer system and the stepper motors, shaft encoders and infra red scanners. The digital circuitry is programmed into a number of Erasable Programmable Logic Devices (EPLDs) using the Altera programming system MAX - Plus 2, more details of which can be found in [48]. The chips are controlled by the root transputer using a sixteen bit external

bus mapped to a particular memory address. The digital logic subsystems therefore appear to the computer as registers in memory, details of the addresses of which can be found in Appendix C. The EPLD's are mounted on cards designed by the Durham University Microcomputer Centre which connect to the transputer boards. The expansion area of the second (non-root) transputer board was used to house a clock chip generating 600kHz and 10kHz clock signals. These clock signals are propagated onto the backplane for use by the EPLD logic.

2.2 Controlling Software Structure

The software which controls the AGV [49] is structured as a number of concurrent processes running on the transputer system. They communicate with each other via 'channels' which can either be memory locations in the case of processes running on the same transputer or hardwired links between different processors. There are twenty such user defined processes running on the computer system, varying in complexity from the fuzzy logic rule interpreter to simple buffer processes. Fifteen of the processes are concerned with the operation of the infra red scanner based location system and the five others form the hierarchical control system for the vehicle for which the basic loop has been shown in Figure 1.4.

The interconnection of the main processes, via the channels used during a test run, is shown in Figure 2.5 while a more detailed breakdown of the way the localisation system processes are arranged can be seen in Figure 2.13. Additional channels exist for the passing of startup information and error messages but they are idle during a run. The following sections cover the tasks performed by each process in more depth.

In brief the functions of the processes are as follows. The message handler receives the startup information from the host PC and initialises the other processes. During execution it stores messages from the other processes and downloads these to the PC when the test is over and the PC is reconnected. The controlling process activates every 200ms, gets the current position from the odometry process, passes this to the fuzzy rule interpreter which processes it and produces a demand steering angle and velocity. This is passed on to the motion control process which translates these demands into step speeds for the motors, within the physical limits of the vehicle's capabilities. The location system controller operates independently taking fixes on the beacons and passing this information on to the

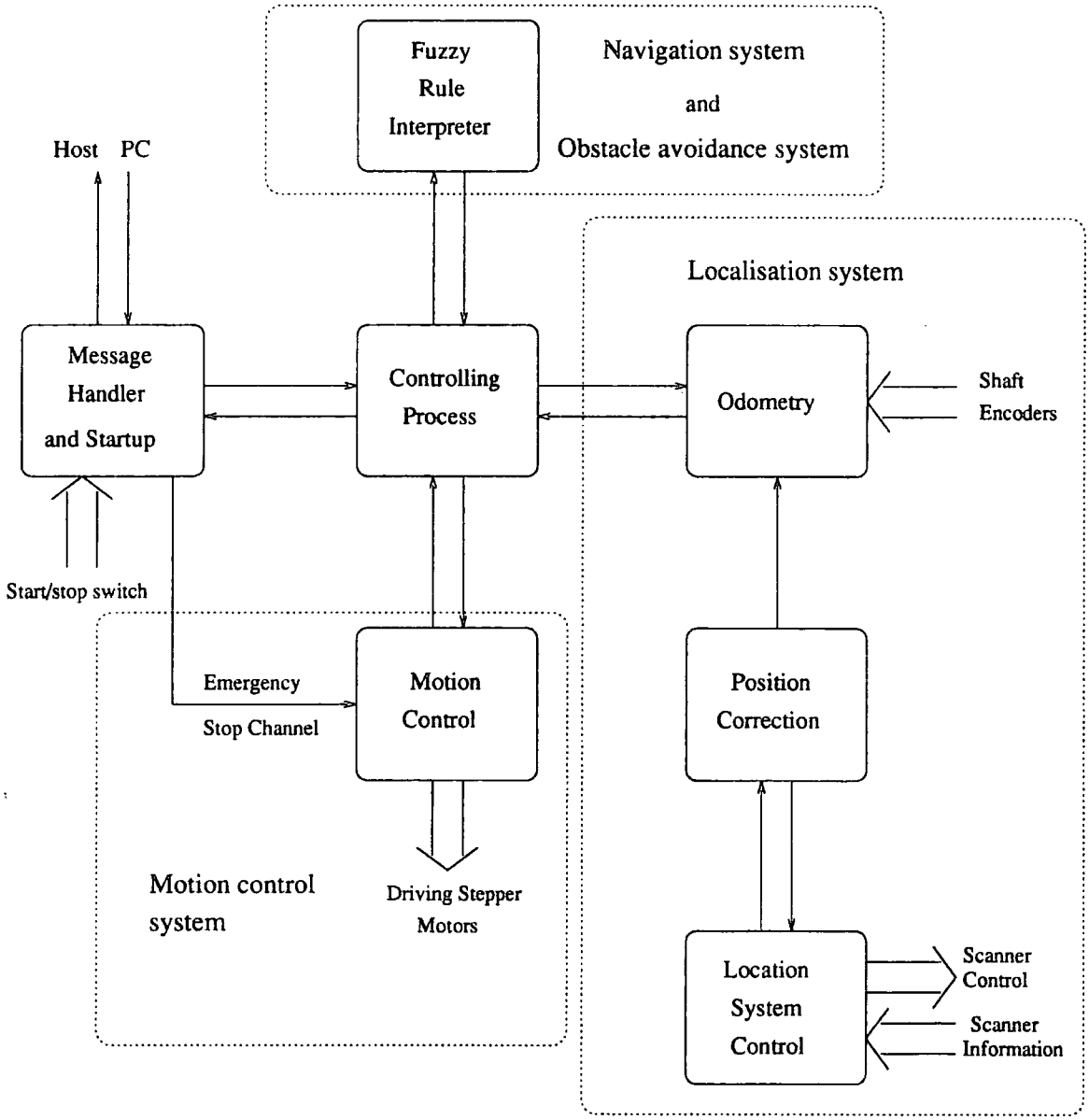

Figure 2.5: Primary Control Processes

position correction routine. This routine attempts to correct odometric errors by utilising the beacon information and passes the necessary corrections on to the odometry process. The odometry process keeps an up to date estimate of the location of the vehicle by using the information from the shaft encoders on each of the rear wheels.

2.2.1 Message Handler

The message handling process, which is connected to the host PC, is responsible for starting up all the other processes by passing them the appropriate data for the test run which is obtained from files held on the PC and for storing messages about the progress of a run. The process has a message input channel from every other process [1] and during a test run the information received over these channels, which takes the form of text messages, are stored as a dynamically allocated list. At the termination of a run the process waits until the start/stop switch on the rear of the vehicle is closed, signalling that the link to the PC has been reconnected. The process then tests the integrity of the link by sending a dummy message before writing all the messages received during a run to the hard disc of the PC.

This process is also used to provide an emergency stop feature and debugging information. If the start/stop switch is closed during a run then the message processor signals to the motion control process to stop the vehicle instantly by switching off the stepper motors. A stop can also be caused by a long period without any messages, indicating that a deadlock situation may have occurred, usually due to an error in one process preventing it responding as expected. The message processor downloads all messages received up to that point to the PC so that the error can be traced.

2.2.2 Controlling Process

The controlling process samples the position, orientation, speed and apparent steering angle of the vehicle at regular sample intervals (currently every 200ms) by interrogating the odometry process. This information is then passed to the fuzzy rule interpreter process which responds to this information by calculating new demand values for velocity and steering angle. The controlling process relays this information to the motion control process

[1] Except for the buffer processes

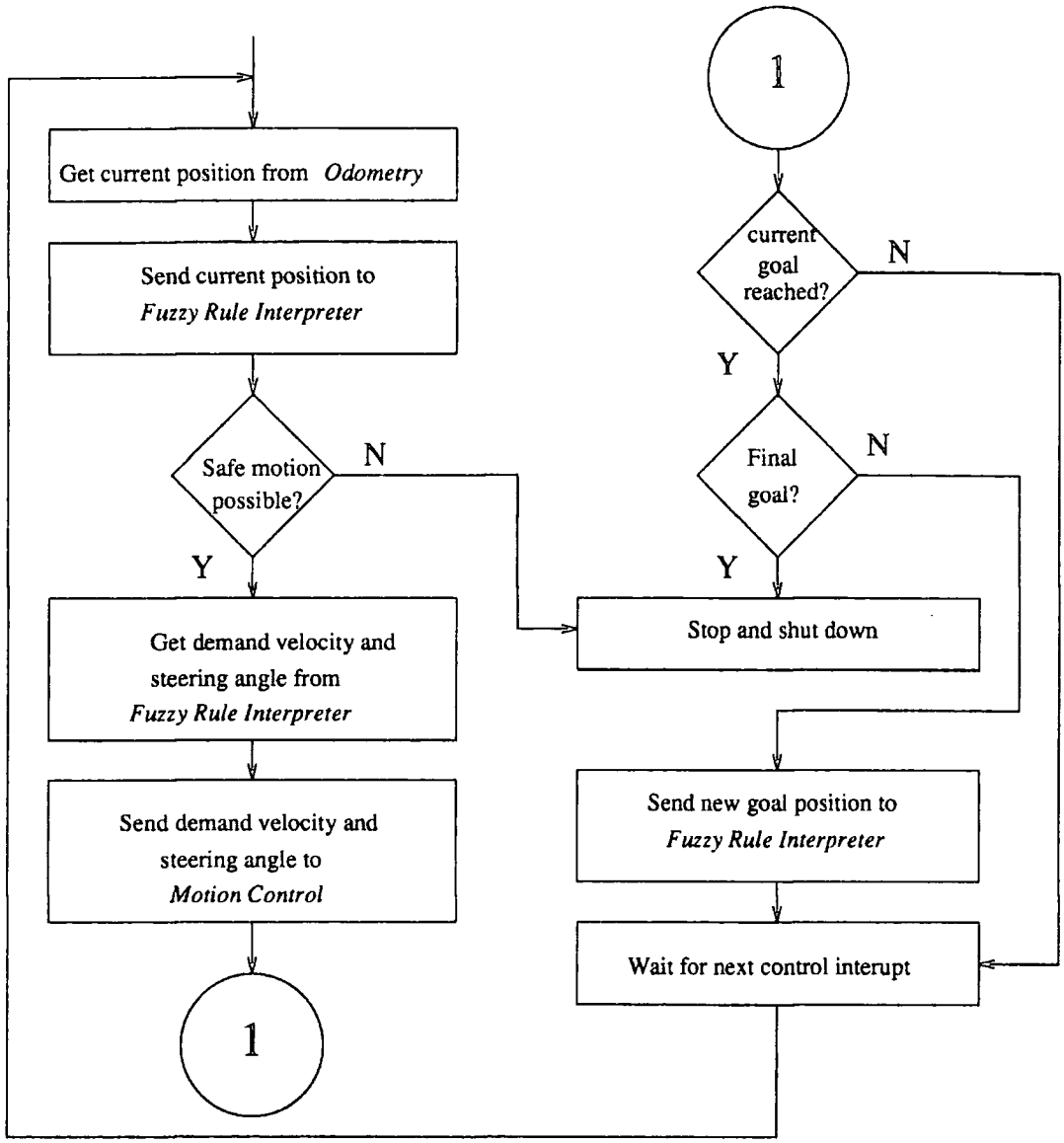

Figure 2.6: Flow Chart of Actions Taken Every Sample Interval.

as a demand velocity and radius of curvature. This process drives the stepper motors to try and achieve these settings. A message is then sent to the message processor informing it of the current state of the vehicle. The controlling process is also responsible for assessing whether or not the vehicle is sufficiently close to the goal location for the run to be over and the vehicle shut down. A flow diagram showing the sequence of events each sample interval is shown in Figure 2.6.

This process therefore is the link between all of the vehicle's systems. It would also have a link further up the hierarchy to path and mission planning levels if they were present in the implementation. It has the capability to change the vehicle's goal position as an additional guide to navigation as described in Chapter 7. The fuzzy navigation and obstacle

avoidance systems are described in later chapters but the following sections describe the software processes used for the motion control and localisation systems along with their associated interface electronics and hardware.

2.3 Motion Control

The function of the motion controller is to produce the necessary control signals to the stepper motors which drive the vehicle so that it moves in the manner demanded by the higher level control systems. It receives demands for the vehicle's velocity and steering angle. The higher level control systems assume the vehicle has either a single, central, steered front wheel or two steered front wheels which act so as to produce the same effect. This configuration was chosen because it more accurately represents vehicles in use for warehouse-type applications and because the research vehicle with its independently driven rear wheels could easily produce the same effects as this configuration. The motion control system therefore has to make the stepper motors behave as if the vehicle was being steered by the front wheel in the manner demanded and also apply limits to ensure the vehicle operates within its capabilities.

A block diagram of the way the controller operates can be seen in Figure 2.7. The demand velocity and steering angle are provided by the controlling process from the result of the navigation and obstacle avoidance systems. These are converted into a demand base velocity, the velocity of the centre of the wheelbase, and a demand steering constant, K_{steer}, which is obtained from Equation 2.3, derived along with the other equations of motion in Section 2.4. These demand values are operated on by the limit loop. This is a local control loop which operates every 100ms. It is used to limit the accelerations required to achieve the demand settings so they are within the capabilities of the vehicle and is covered in Section 2.5. The values output by the limit loop are converted to count values which are written to the registers in the motion control EPLD which provides the variable frequency square wave to the stepper motor driver cards which operate the motors.

The speed of the stepper motors is controlled by dividing down the frequency of the 600kHz clock using a counter and latch. Each stepper has an associated latch which holds the start value for the count. On a signal from the computer the counter is loaded with this latch value and the counter enabled. The counter counts down to zero and then gives out a

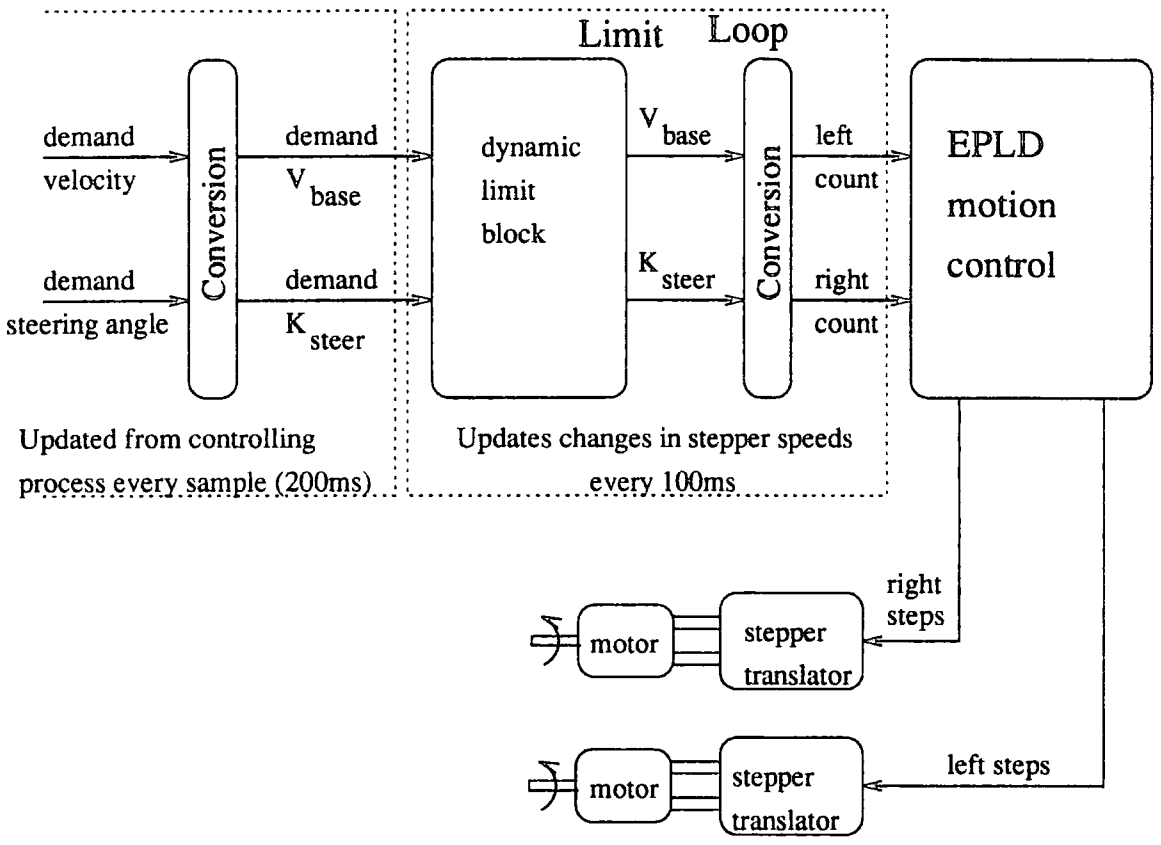

Figure 2.7: Block Diagram of Motion Control System

pulse to the stepper translators, at the same time it loads the latched values and starts the counting sequence again. This provides a constant stream of pulses to the stepper motors which may be controlled in frequency to control the speed of each wheel. This makes the control from the transputers very simple and removes the necessity of generating the stepper driving pulses in software. The motion controlling chip can also produce a signal to sound the buzzer on the AGV.

Descriptions of the logic, written in the Altera Hardware Description Language (AHDL) can be found in [48]

2.4 Equations of Motion

This section derives the equations used in the various stages of the motion controller described above. For a given steering angle (ϕ) the vehicle will travel on a curve, as is shown in Figure 2.8 the centre of the circle which the vehicle follows is at the intersection of the perpendiculars to the steered front wheel and the fixed rear wheels. Since the point of

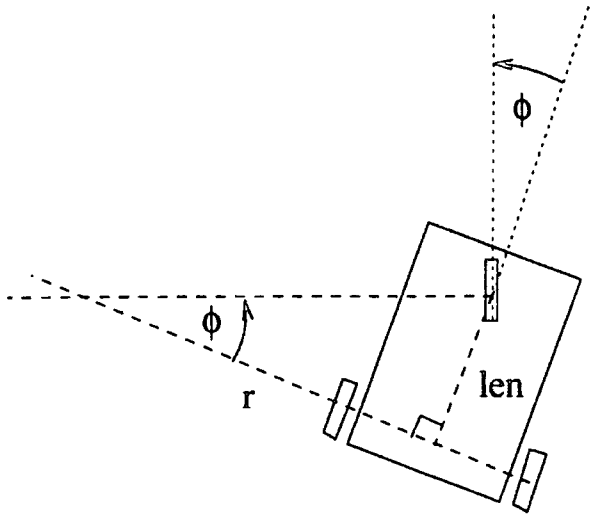

Figure 2.8: Diagram Showing Derivation of Turning Radius From Steering Angle

reference used to signify the position of the vehicle is the centre of the rear wheel base the radius of curvature (r) can be found from Equation 2.1 where len is the distance between the centres of the front and rear wheelbases. This equation gives negative values of r for negative steering angles. This sign component indicates whether the turn is to the right (+ve r) or the left (-ve r).

$$r = len/\tan\phi \tag{2.1}$$

K_{steer} is the proportion of additional distance which the left wheel will travel, relative to the centre of the wheelbase, on a given radius of curvature r and is found from Equation 2.2 where W_b is the wheelbase of the AGV. It can be derived directly from the demand steering angle by Equation 2.3.

$$K_{steer} = \frac{W_b}{2r} \tag{2.2}$$

$$K_{steer} = \frac{W_b}{2}\frac{\tan\phi}{len} \tag{2.3}$$

The required wheel velocities v_{left} and v_{right} can then be found from the following equations.

$$v_{left} = v(1 + K_{steer}) \tag{2.4}$$

$$v_{right} = v(1 - K_{steer}) \tag{2.5}$$

Each wheel has its own counter which will count for n cycles of the clock before a step is demanded from the wheels. The required count value for a wheel to achieve a velocity of v is found from the following equation.

$$n = \frac{\pi d G_r f_c}{v} \tag{2.6}$$

Where:

d = Diameter of wheel (20cm)
G_r = Gearing ratio (7/36)
f_c = Clock frequency (600kHz)

2.5 Velocity and Steering Angle Limits

The acceleration of the vehicle and the rate at which it can change steering angle are limited in practice by the performance of the stepper motors. If the torque demand upon the stepper motors is too high then the motors will stall. The available torque depends upon the speed at which a motor is operating. To ensure that the stepper motors avoided a stall the change in velocity of each wheel would need to be limited depending on the position of the motor on its torque speed characteristic. This problem is specific to the research AGV but clearly in a more conventional vehicle there is a limit on the acceleration which can be supplied and a limit on the rate at which the angle of the steered wheels can be changed. This limit is also present in the AGV since the front castor wheel opposes fast changes in direction.

To prevent stalling of the stepper motors and, to some extent, mimic a more conventional steering mechanism, limits were placed on the rate of change of steering angle and the base acceleration of the vehicle. The torque demand depends on the linear and angular acceleration of the vehicle. The angular velocity, the rate of change of the vehicle's orientation (θ), can be found from the radius of curvature (r) by applying Equation 2.7. The angular acceleration is found by differentiating this to give Equation 2.8.

$$\frac{d\theta}{dt} = \frac{V_{base}}{r} = \left(\frac{2}{W_b}\right) V_{base} K_{steer} \tag{2.7}$$

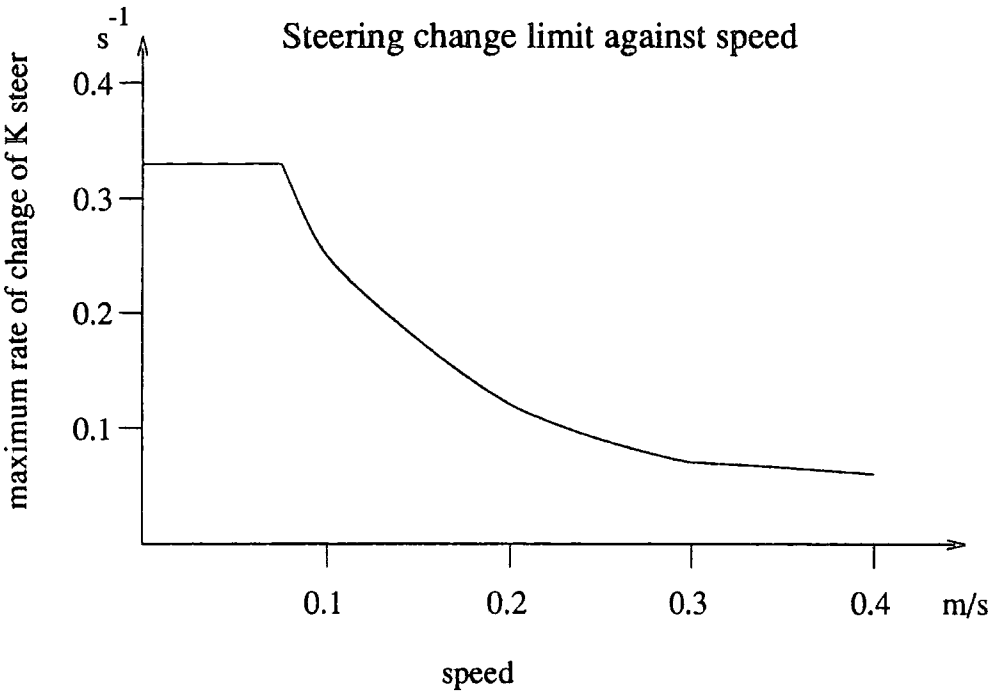

Figure 2.9: Maximum Rate of Change of Steering Constant Against Speed.

$$\frac{d^2\theta}{dt^2} = \left(\frac{2}{W_b}\right)\left(K_{steer}\frac{dV_{base}}{dt} + V_{base}\frac{dK_{steer}}{dt}\right) \tag{2.8}$$

It can be seen that in order to limit angular acceleration the maximum rate of change of the steering constant should depend on the base velocity and the allowed acceleration should depend on the current steering constant. The steering constant K_{steer} is linked to the steering angle by Equation 2.3 so any limit on its rate of change corresponds to a limit on the rate of change of steering angle.

The limit on the rate of change of steering constant is found from Equation 2.9 and can be seen graphically in Figure 2.9. The limit is fixed for velocities up to 0.075 m/s and then is reduced for higher velocities. This prevents unrealistically fast changes in steering angle being made at very slow speeds, and traps the potential division by zero if the vehicle is stationary.

$$\frac{dK_{steer}}{dt} = \begin{cases} 0.333 & \text{if } |v| <= 0.075 \\ 0.025/|v| & \text{if } |v| > 0.075 \end{cases} \tag{2.9}$$

The limitations placed on the acceleration of the vehicle are more complex. To start with the controller only allows a fifth of the difference between the demand and actual

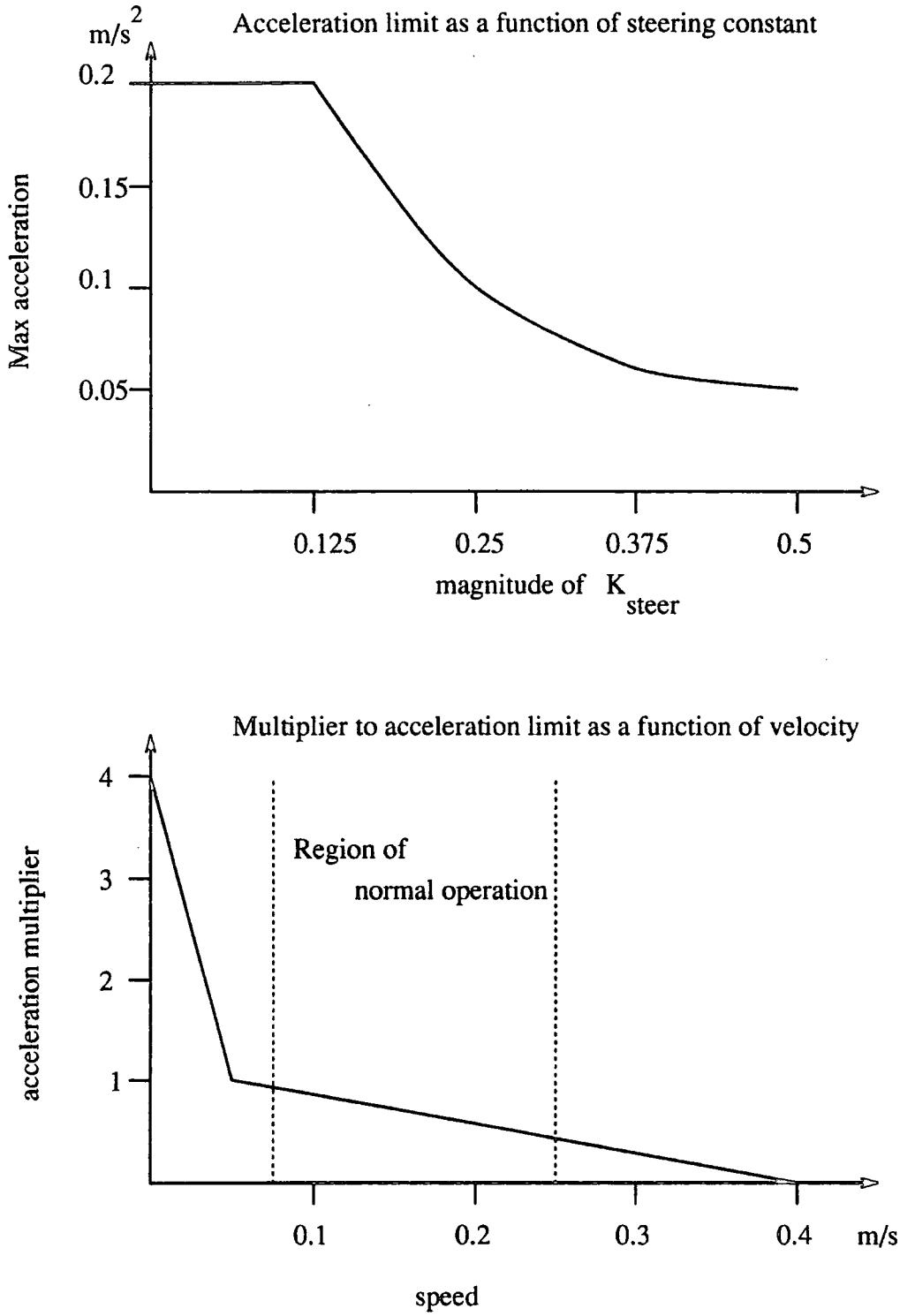

Figure 2.10: Limits Placed upon the Vehicle's Acceleration.

velocities to be achieved during a control interval. This ensures a smooth reduction in acceleration as the desired speed is approached. As described above it is necessary to reduce the acceleration permitted at large steering angles depending on the value of the steering constant to limit the angular acceleration. This is done by applying Equation 2.10 which gives a fixed ceiling for small steering angles but falls off rapidly for sharper turns.

$$\frac{dv}{dt} = \begin{cases} 0.2K_v & \text{if } |K_{steer}| <= 0.125 \\ 0.0025K_v/|K_{steer}| & \text{if } |K_{steer}| > 0.125 \end{cases} \tag{2.10}$$

For deceleration

$$K_v = -2$$

For acceleration (2.11)

$$K_v = \begin{cases} 4 - 60|v| & \text{if } |v| <= 0.05 \\ 1.15 - 2.85|v| & \text{if } |v| > 0.05 \end{cases}$$

The constant K_v is determined by Equation 2.11 which allows higher acceleration at very low velocities, when the vehicle is accelerating from rest and plenty of torque is available, but has a value of between 0.8 and 0.6 over the normal operating range. This is necessary to take account of the reduction in available torque at high speeds, a common factor of all motors. The curves produced by these equations can be seen in Figure 2.10.

2.6 Localisation System

In order for the control system to be thoroughly tested the AGV needs to have a means of locating its position in the laboratory. Over short distances this can be done by monitoring the rotation of the wheels, as described in the following section. Unfortunately the errors in this odometric measurement increase with the distance covered so it is important to have a system capable of correcting these errors. This is provided by a set of infra red beacons positioned around the room which three scanners located on top of the vehicle lock on to and the measurement of the angles to the beacons is used to triangulate the position of the AGV. This beacon location system is covered in more detail in Appendix B but a brief

description is given in Section 2.6.2.

2.6.1 Odometry

The odometry process keeps the most up to date estimate of the vehicle's position and orientation as well as maintaining estimates of the average velocity and steering angle over the past few centimeters.[2] The current position can be requested by one of several processes, including the controlling process and several of the scanner control processes. To be able to satisfy requests from many different sources the process is designed to run asynchronously and calculates the current position each time it receives a request and then sends this to the requesting process.

The shaft encoders mounted on each of the rear wheels produce two square waves in quadrature, of 500 pulses per revolution and a single reference pulse per revolution. These signals are fed into an EPLD programmed to process this information. It includes an up/down counter for each wheel that counts the pulses. Its AHDL description can be found in max2work\locnbrd\odom.

When the odometry process receives a request for the current position it reads the current values of the counters associated with the shaft encoders on each of the rear wheels and uses the differences in counter values to calculate how far each wheel has travelled. This information is used to calculate the change in position and orientation of the vehicle based on the assumption that the distances measured correspond to the vehicle moving along a single curve segment with no wheel slip occurring. The distance travelled by each wheel is found from Equation 2.12 where n is the difference in counts between the current and last reading of the counter and d is the diameter of the wheel. Care is taken to ensure that this difference is accurately maintained when the counters 'roll over' in either direction.

$$\text{distance} = \frac{n\pi d}{500} \tag{2.12}$$

From this it is possible to find the average distance travelled $\bar{d}$, the change in orientation

[2] currently 3.5cm

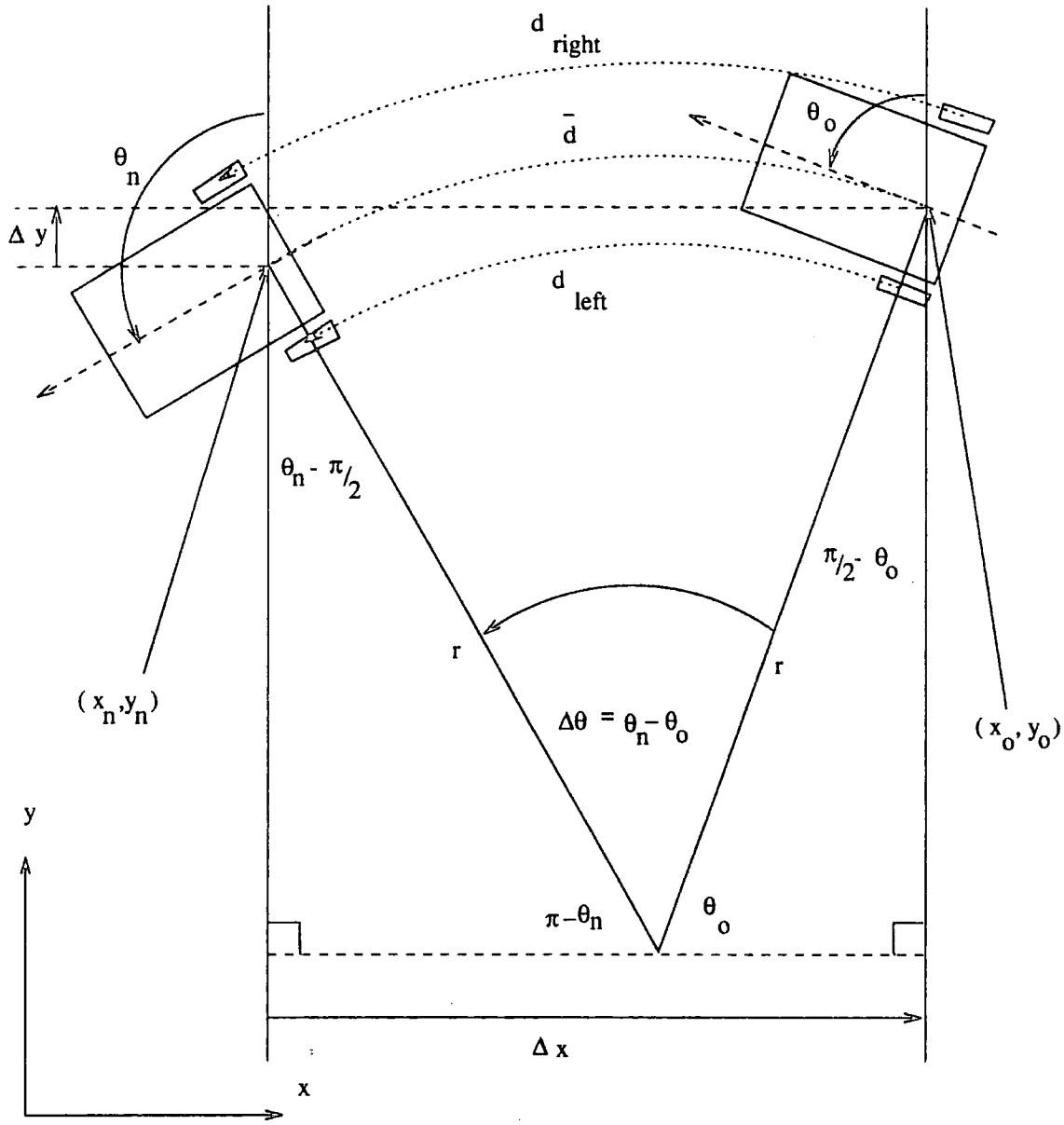

Figure 2.11: Change in Vehicle Position, Based on Odometric Measurements

$\Delta\theta$ ($\Delta\theta = \theta_n - \theta_o$) and the radius of curvature r so long as the wheelbase W_b is known.

$$\bar{d} = \frac{d_{left} + d_{right}}{2} \tag{2.13}$$

$$\Delta\theta = \frac{d_{left} - d_{right}}{W_b} \tag{2.14}$$

$$r = \frac{\bar{d}}{\Delta\theta} \tag{2.15}$$

The position is measured from the midpoint of the rear wheelbase relative to a global origin, situated in one corner of the laboratory.

The situation is shown in Figure 2.11, which shows the vehicle moving with a constant curvature of r over a distance $\bar{d}$. The vehicle starts at (x_o, y_o) with orientation θ_o and ends at (x_n, y_n) with orientation θ_n. The change in position is found using equations 2.16 and 2.17.

$$\Delta x = \begin{cases} r(\cos\theta_o - \cos\theta_n) & \Delta\theta \neq 0 \\ \bar{d}\cos\theta_o & \Delta\theta = 0 \end{cases} \tag{2.16}$$

$$\Delta y = \begin{cases} r(\sin\theta_n - \sin\theta_o) & \Delta\theta \neq 0 \\ \bar{d}\sin\theta_o & \Delta\theta = 0 \end{cases} \tag{2.17}$$

These changes are used to update the current position estimate which is communicated via a return channel to the requesting process.

This estimate can be changed by corrections sent from the position correction process based on the results of the beacon location system. This occurs when a message is received from the position correction process containing the values by which the position estimate currently being used by the odometry process should be adjusted in x,y and θ (vehicle orientation).

Estimates of velocity and apparent steering angle are also maintained so they can be fed back into the control system. Since the odometry process updates information about the current position when requested it is possible for the time difference between requests to be only a fraction of a second. Using this small time difference would introduce unacceptable quantisation errors as the shaft encoders might only have counted one or two pulses. The estimates of velocity and steering angle are therefore only updated when the vehicle has travelled at least 3.5cm. The velocity is measured by dividing the average distance travelled by the wheels by the time since the estimate was last updated. The estimated steering angle (ϕ) is found from Equation 2.18 where len is the distance between the centres of the front and rear wheelbases (front wheel and centre of rear wheelbase in the case of the AGV). This equation can be derived by substituting Equation 2.1 for r in Equation 2.15.

$$\phi = \arctan\left(\frac{\Delta\theta\ \text{len}}{\bar{d}}\right) \tag{2.18}$$

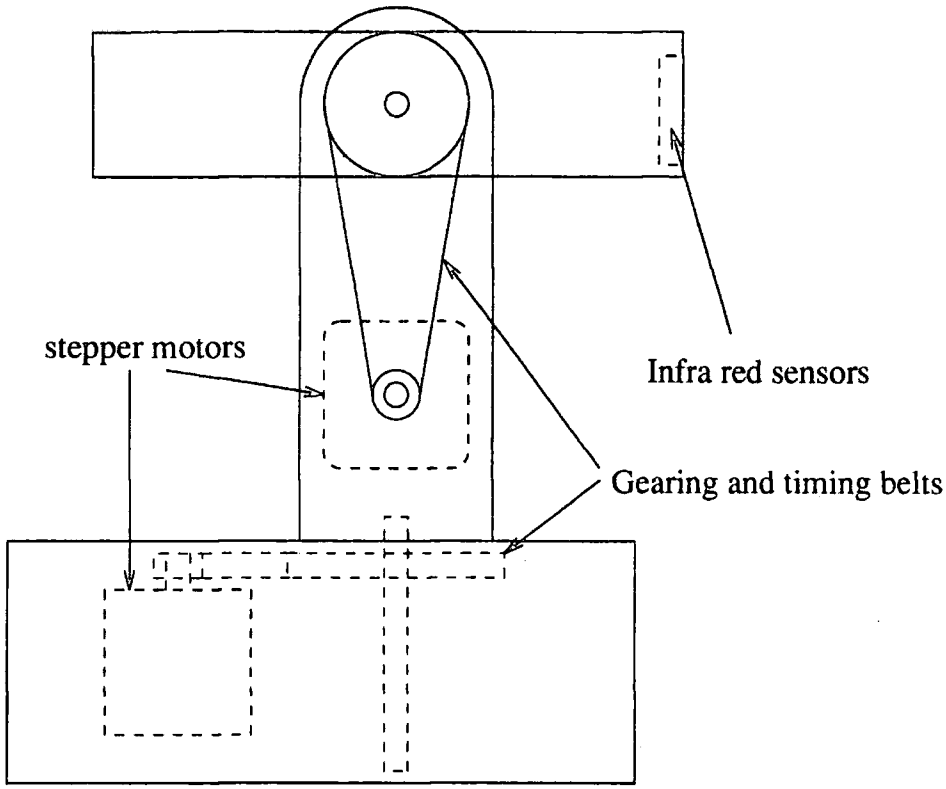

Figure 2.12: Scanner Incorporating Infra Red Sensors

2.6.2 Beacon Location System

The beacon location system locates the position of the vehicle by triangulation from known infra red transmitting beacons. More details about this method can be found in Appendix B. The angles from the vehicle to beacons is measured by infra red receiving scanners. A scanner is shown in Figure 2.12 and can rotate the tube containing the infra red receptors in both horizontal and vertical planes using stepper motors. Each scanner is controlled by logic programmed into an EPLD. This enables the computer to drive the scanner directly, cause the scanner to move to its calibration position or track an already acquired beacon. Two registers are maintained on the EPLD, containing the number of steps taken by the stepper motors so that the computer system can read these to determine the angles to the beacon.

When the vehicle starts a run all three scanners are calibrated and then assigned a beacon to track. An independent control system running on the transputers handles problems such as recalibrating the scanners if steps are missed, preventing their cables from becoming entangled and assigning new beacons if the scanner loses a beacon due to extreme range or an intervening obstacle.

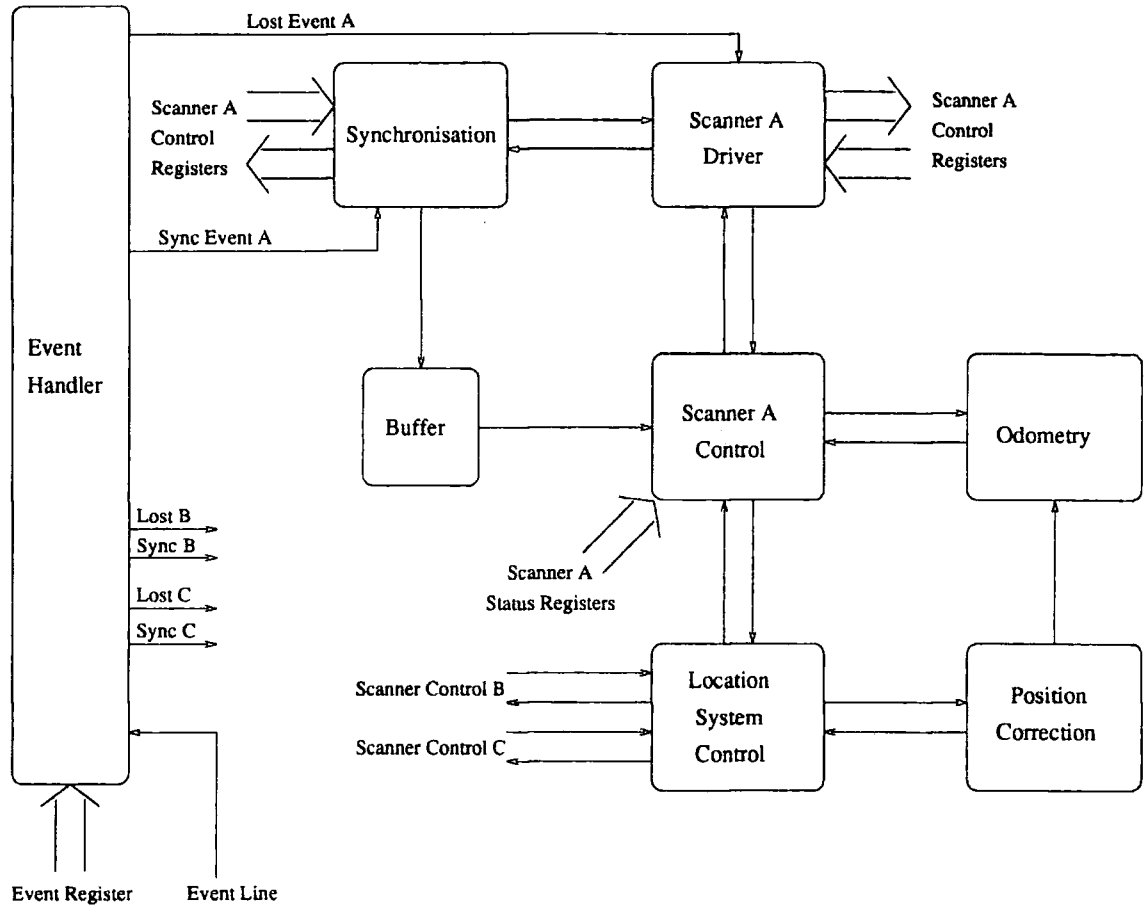

Figure 2.13: Location System Control Processes

2.6.3 Location System Control

Figure 2.13 shows how the location system control processes are connected. Each scanner has four processes which are associated with it and there are three common processes, the event handler, location control system and position correction processes. The event handler is alerted when the event line of the transputer is asserted (goes low). This happens when any scanner passes one of its calibration optical slot switches or a scanner either acquires or loses a beacon. The event handler reads the event status register and sends messages out on channels to all relevant processes to deal with the event. It then resets the event register.

The location control process acts as a high level supervisor for the scanners. It knows the positions of all the beacons and assigns the scanners to beacons using a simple fuzzy expert system. The scanner control processes are regularly polled to measure the current angles to the beacons and to assess whether or not a new beacon needs to be assigned to a

particular scanner. The angles measured, and the position from which it was thought this was done are passed to the position correction system.

The scanner control process keeps track of the current status of its scanner. When requested by the control system it passes back information on the status of the scanner and the angle it is currently pointing at. It receives messages from the driver process about the scanner when the status changes (such as successfully acquiring a beacon) and it is informed, via a buffered channel, if the synchronisation process has detected a loss of calibration. The scanner controller knows how far its scanner can rotate in each direction before it entangles its cables and can order a 360° turn to untangle the cable if needed.

The scanner driver process attempts to achieve the commands of the controller process by writing to the control registers of the EPLD associated with each scanner. For instance to acquire a beacon it calculates the number of steps necessary to achieve the horizontal and vertical orientation needed to point directly at the beacon and drives the scanner to this position. Once this is achieved it instigates a search pattern around that position until a strong enough signal is found to enable the EPLD tracking logic to follow the beacon. While the beacon is being tracked it looks for a message from the event handler signifying that the beacon has been lost and attempts to reacquire it in case it was a momentary loss (such as is caused by someone walking past the scanner) before alerting the higher level process.

The synchronisation process maintains the calibration of the scanners. When requested to do so it drives the scanners to their calibration positions where two slotted optical switches confirm the scanner's precise position. During normal operation the synchronisation process is informed by the event handler if the scanner passes one of theses switches and if it finds that the scanner does not appear to be at a valid position to have done this it signals to the controller that recalibration is required via the buffer process. It is necessary to use this buffer process to ensure that the event handler and scanner controller processes are never 'blocked'.

Chapter 3

Simulation and Fuzzy Control Software

The AGV controller was developed while using a simulation of the vehicle and then tried out in practice once the controller performed satisfactorily in simulation. There are several reasons for taking this approach, firstly because tests can be run at several times real speed, secondly to prevent errors causing damage to the vehicle and thirdly so that the inner workings of the control system can be examined during simulated test runs.

It is, however, important to use a real vehicle since many problems are present on a real vehicle which are not apparent in a simulation. These include the effects of noise, sensor inaccuracies and dynamic limits on performance. The importance of using a real vehicle has been underlined by many researchers. Payton et al [37] when discussing their experiences on the Autonomous Land Vehicle (ALV) project say of their initial approach 'On paper and in simulation, it seemed that this approach would be effective,' but go on to concede 'when we attempted to perform this mission with the ALV, the deficiencies of our method became strikingly clear.'

One of the important functions of the simulation is to remove problems from the control system before they occur on the vehicle. For this reason the software for the control system was developed in such a way that it could be moved from simulation to the vehicle with only minimal changes. In this chapter the overall structure of the simulation program is presented and its capabilities described. The main sections of the program, data initialisation, fuzzy

controller implementation, vehicle simulation and information display are described in the following sections. The fuzzy control routines described here as part of the simulation are identical to those used on the vehicle itself. Readers who do not wish to know further details about the inner workings of the simulation program may ignore the rest of this chapter or leave reading it to a later date. In particular the reader may have to refer forward to Chaper 4 to get an idea of the size and nature of the rules and set definitions in order to understand the reasons behind their implementation. The ideas behind the controller are contained in the following chapters and ignorance of the methods used to implement the controller will not prevent the reader understanding the theory behind it.

3.1 The AGV Simulator

The simulation of the AGV was developed from an earlier program written to mimic the solution to the truck backing problem applied by Kosko [39] in which a fuzzy ruleset controls a simulated truck backing into a parking space. Following the experience gained in producing this program the software was developed first to control an AGV seeking an arbitrary docking position and orientation and then to do this while avoiding obstacles. The program makes use of a large number of data files to enable changes to be made to the test being run and the controller without needing frequent recompilations of the code. This was essential since the main purpose of the simulator was to act as a tool for controller development. In addition to giving a graphical display on the screen the program can save data from each step of the run in a number of different formats to allow comparison, using graph drawing packages, of different starting positions and controller settings. The 'C' code for the simulation program has been bound into a separate report [50] but the functionality of the program is described here.

Figure 3.1 shows the sequence of operations in the main routine which controls the operation of the simulation. The first part of the routine sets up the data structures and reads a data file, specified on the command line, containing information on the test to be carried out. The *Setup* routine is then called which reads in all the information on the rules to use and obstacle positions from data files and sets up the graphics screen. This process is covered in the next section of this chapter.

The main loop simulates the actions which are taken by the main control routine on the

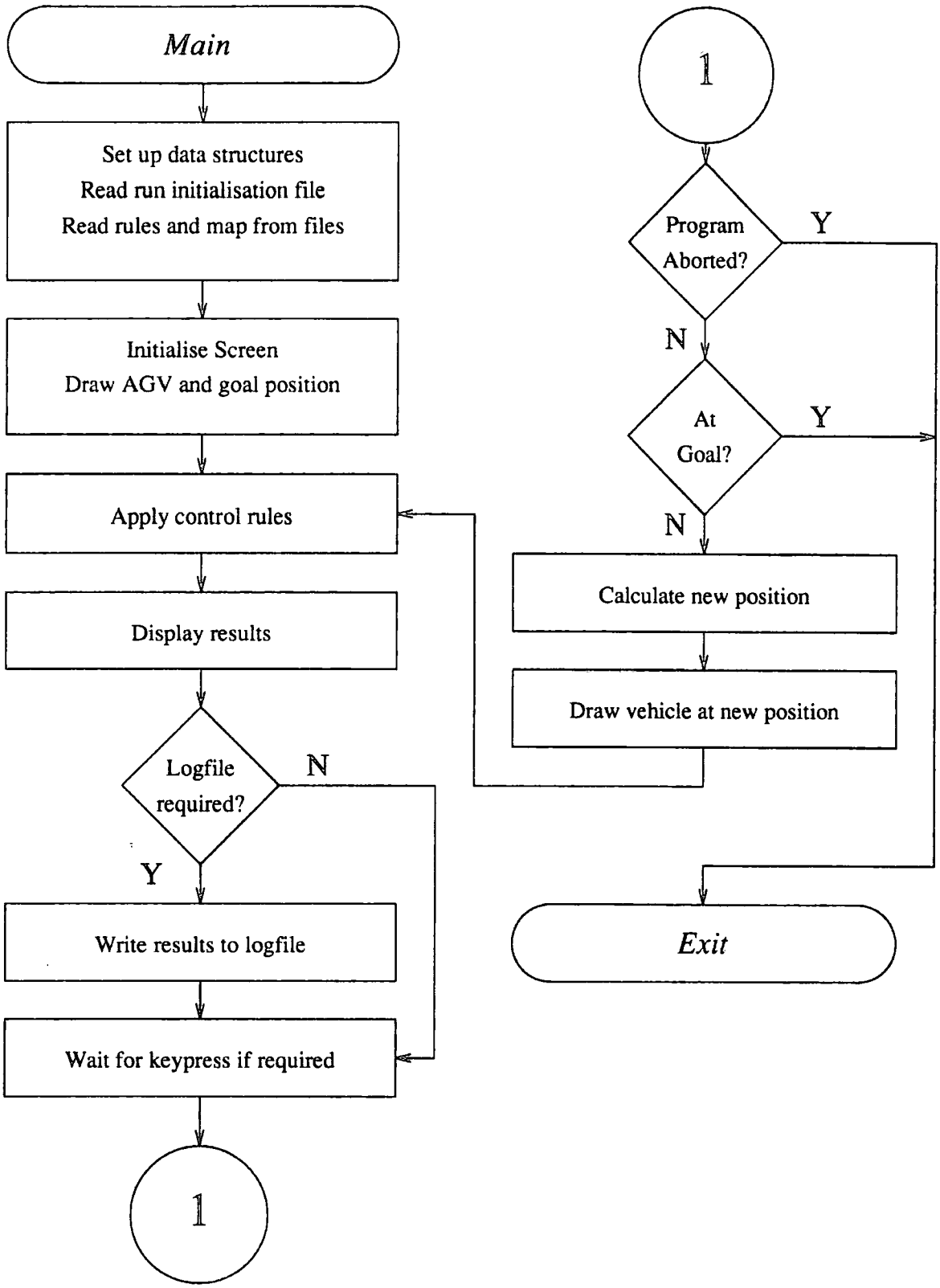

Figure 3.1: Control Flow Through the Main Simulation Loop

vehicle. The AGV is displayed on the screen at its current position and the goal position redrawn if the vehicle is close enough to it to cause overlap.

The control routines are then called to determine the correct steering angle and velocity to be applied, based on the output of the fuzzy logic controllers. The software which implements these controllers is covered in Section 3.3. During execution these routines display information about the rules which have been applied on the right hand side of the screen so that any incorrect rules can be identified.

The display is updated and a check performed to see that motion without collision is possible. The position and orientation of the AGV along with the demand and actual velocity and steering angle are written to a file, if this option was selected at the start, and the program waits for a keypress. Following this a check is performed to see if the AGV is within allowed tolerances of the goal position and orientation. If multiple goals are being used the goal position is changed to the next in the list but if the final goal has been reached the loop is exited. The loop is also exited if no motion is possible without a collision or if the 'q' key has been pressed. Otherwise the AGV is cleared from the screen, its position updated depending on speed and steering angle as described in Section 3.6 and the loop repeated.

3.2 Initialisation Routines

Much of the information used by the program is held in data files to allow it to be changed without altering and recompiling the main program. This section first introduces the data structures used by the program and then goes on to describe the routines found in the file `Setup.c`. These routines read information from data files and store the information in memory in a suitable format for use by the program. The data files include the definition of the FAM[1] bank matrix, definitions of the set boundaries for the obstacle avoidance sets and the initial condition of the occupancy grid. Additional tasks such as setting up the graphic output screen and calculating the normal distribution to be used are also performed by these routines although the graphic functions are not documented in detail.

[1] Fuzzy Associative Memory

3.2.1 Data Structures

There are several important data structures used by the program which are defined in the `Define.h` header file and allocated memory at the start of the program. Arrays are used to store the fuzzy set definitions, the FAM bank matrix linking the input and output sets and the linguistic labels for the input and output sets. More complex data types are defined which link together several arrays.

The 'C' structure *fams* contains the definitions of all the fuzzy input sets, the number of sets for each state variable and the rule matrix. This allows all the information used for the navigation function to be accessed from one structure. A second 'C' structure *avoidarea* links together all the information needed to define an obstacle avoidance set. In addition to arrays of the x,y positions of the set boundaries this structure holds the number of the output set which the rule inhibits and how much inhibition should be applied if the set is activated. All the obstacle avoidance rules are contained in an array of these structures which is called *obset*.

The velocity rules are held in a structure *velocity_rules* which holds entries for the first condition which must be satisfied for the rule to apply, the consequent output set number and a pointer to the next rule. Each condition is itself defined as a structure which holds a pointer to the fuzzy set definition it refers to, the number of the state variable it is associated with, a flag indicating if the rule is to be inverted and a pointer to the next condition.

The fuzzy sets are defined by four values which represent the vertices of the trapezoidal sets. The first value is the lowest possible number which has a non-zero membership of the set, the second value is the lowest number which has a membership value of 1 while the third and fourth numbers are the highest values to have memberships of 1 and 0 respectively. Each set has a corresponding two letter label which represents the name of the set. For example 'NP' is the label for a Negative Perpendicular angle.

The FAM matrix or rule table is held in the integer array *rules* which is accessed from within the *fams* structure. The array is three dimensional and has an entry for all possible combinations of input sets for the three state variables. The entry in the array is the number of the consequent output set for the rule. A value greater than the total number of output sets is taken as indicating there is no corresponding rule.

Finally a structure which is given the type name VEHICLE is defined to hold the current state of the AGV. This structure holds the x,y position of the vehicle, its orientation, current velocity and steering angle. This structure is frequently passed between routines which need to know the current state of the vehicle.

3.2.2 Setup Routine

The data structures are set up within the main program and a set up file for the test, specified on the command line when the program is run, is read to find the start and goal positions and orientation, the names of the data files to be used in setting up the controllers and the values of various other options such as the controller sample time. With this information the routine *setup* calls all the various lower level routines to initialise the data structures ready for execution of the program. First the FAM bank matrix is read in from a file, then the obstacle set definitions and finally the initial occupancy values of the occupancy grid are also read from files. The names of the files to be used are specified in the set up file for the testrun. A routine is then called to set up the graphics output screen to display the motion of the AGV and the workings of the controller. The last call is to a routine which displays which areas of the screen are occupied according to the values in the initial occupancy grid.

3.2.3 Navigation Rules Setup

The routine *read_fam* reads the data file containing the linguistic rule definitions. This file consists of a series of two letter labels for output sets arranged in a matrix. The position of each label in the file indicates which input sets activate that particular response. The labels used can be seen in Table 4.1 in the following Chapter. With the current settings the program reads in four different tables, one for each range set, each of which it expects to be preceded by a title line. The two letter labels indicating the output set associated with each rule are read in a line at a time. The routine expects as many labels as there are goal heading error sets. Taking symmetry into account there are eight and as many lines as there are heading sets (nine). When a whole line has been read the labels are compared against the labels defined in the program's header files to find which output set number each corresponds to and this is entered in the array defining the rules.

3.2.4 Obstacle Avoidance Set Initialisation

Obstacle avoidance sets are covered in Section 5.4. The *read_obsets* routine initialises the values of these sets from a file. The file is expected to contain definitions of the sets starting with the label for the output set with which it is associated, then the inhibiting value of the set and finally four sets of x,y coordinates defining the boundary of the set. All these entries are separated by a single space with the last entry ending the line with a newline character. The values, except for the label, are all read directly into the array of *avoidarea* structures which hold the rules. The label is read into a buffer and then is compared with the labels defined in the header files to find which output set number it refers to. If no match for the label is found the program is terminated with an error message indicating the location of the fault.

Only the sets corresponding to left hand turns and straight ahead are stored in the data file. If the data is found to refer to a set to the left of the vehicle then a set for right hand turns is generated by symmetry. This simply involves subtracting the associated output rule number from the total number of rules and copying the coordinates to the new set definition with the x ordinate multiplied by -1 to reflect it about the y axis.

3.2.5 Occupancy Grid Initialisation

This routine fills the array holding the occupancy values according to the data held in a file specified in the test run set up information. This file contains 139 lines of 139 characters corresponding to the ten centimeter squares of the grid. Each character may only be one of the characters from the array *key* which is defined in the routine, the position of the character in this array indicates its occupancy value. This enables grids to be generated very easily using a text editor.

The loop for y values starts from the maximum value and is decremented since the file starts with the line furthest from the origin. When each character is read the routine checks for an early end to the line or the file and exits with an error message if this occurs. The character read is then compared against the array *key* to find its position. This key consists of '.' representing zero occupancy and the numeric characters representing the percentage occupancy. '1' represents 10% occupancy up to '0' representing 100% occupancy. If no

match is found an error and its position in the file is reported and the program execution terminated. Otherwise the occupancy value indicated is placed in the array which contains the occupancy grid.

3.3 Fuzzy Logic Software

The vehicle makes use of fuzzy logic controllers to control both the velocity and steering angle of the vehicle. The development of the controllers is described in Chapter 4 but the following sections describe the software routines which apply the controller rules. Most of the routines described here implement standard fuzzy logic structures as described in Appendix A but other routines apply the new techniques developed as part of the research which are introduced in Section 5.7. These new techniques operate to enable the combination of the fuzzy fit vector from the navigation controller with a *mask vector* produced by the obstacle avoidance routine. The routines which generate this mask vector are described in Section 5.5.

While the same defuzzification routines are used for both controllers the natures of their rule bases make the structure of the fuzzification routines slightly different. In the case of the navigation controller the rule base is fully populated and consists of hundreds of rules (540 in the final design). The most efficient way to access this is to have a matrix holding the rules so that the rules which apply to any given situation can quickly be found without searching the entire rulebase. In the case of the velocity controller there are only a few rules and so they are most efficiently stored as a list.

The flow diagram Figure 3.2 shows the routines used to implement the fuzzy logic controllers. The routines themselves are described in the next section. The first stage is the routine which calculates the state variable values used for the navigation controller from the vehicle's position and orientation relative to the goal. The fuzzy rules are then applied to give a fuzzy fit vector indicating the level of desirability of the steering angle sets which is spread by the next routine to make it ready for combination with the mask vector. Following this the obstacle avoidance controller is consulted, if it is active, and this generates the mask vector. The fit and mask vectors are combined and put through the windowing function before the defuzzification routine is applied to give the output steering angle. The velocity controller which follows this is a standard fuzzy controller and after

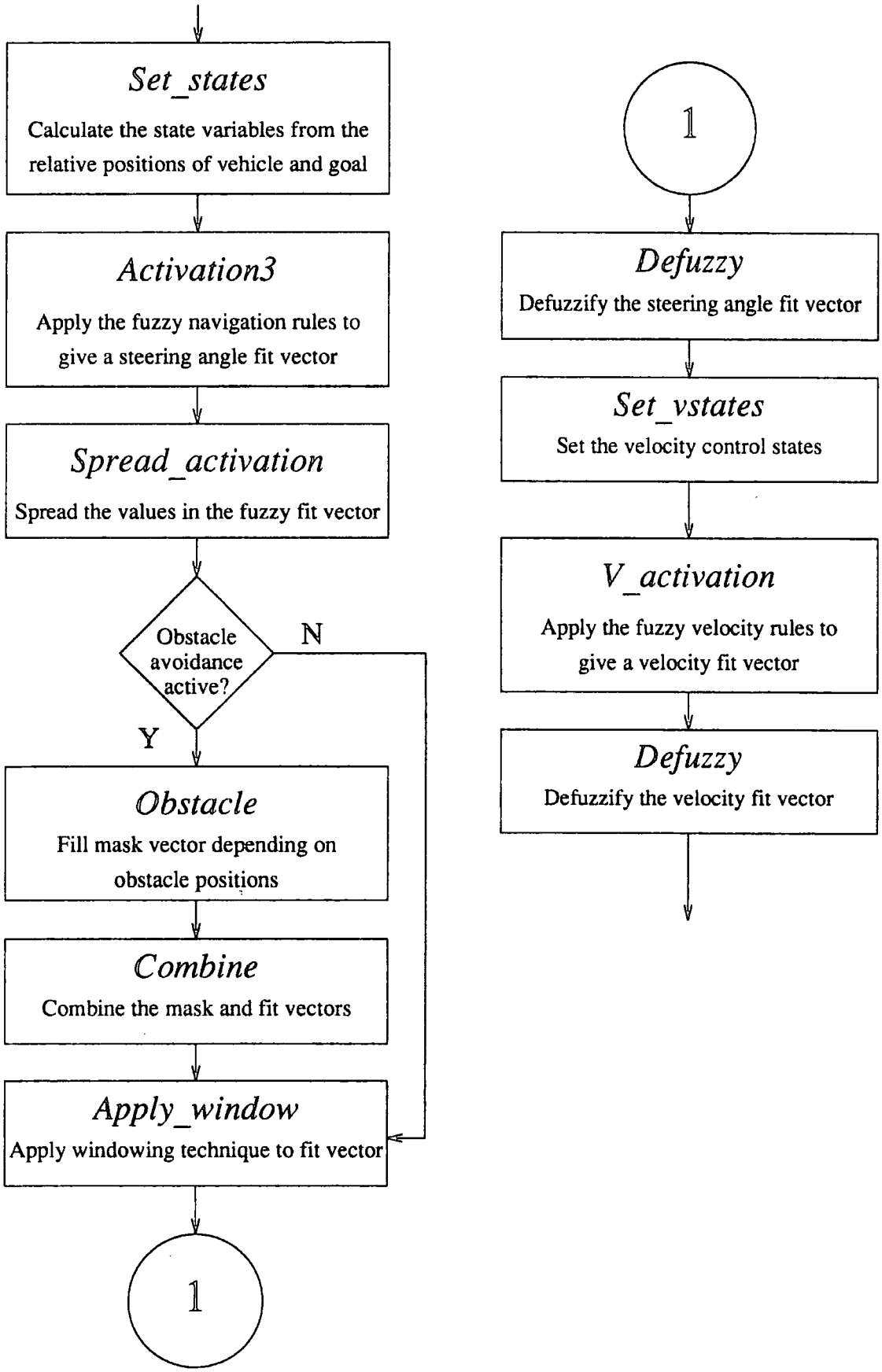

Figure 3.2: Control Flow Through the Fuzzy Logic Routines

calculation of the state variable values the rules are applied and the resulting fit vector defuzzified to give the demand velocity.

3.4 Fuzzification Routines.

This section describes the routines found in the files `Fuzzy3.c` and `Vfuzzy.c` which generate the output fuzzy fit vector by fuzzifying the input state variables and applying the rules contained in the FAM bank matrix for the controller.

3.4.1 Input Fuzzification

The first stage in any fuzzy controller is the fuzzification stage where the input variables are converted into membership values of fuzzy sets defined over the input universe of discourse. The function *fuzzyval* finds the membership of an input variable within a trapezoidal set. The choice of trapezoidal sets makes the definition and evaluation of fuzzy sets simple while enabling the use of both symmetric and non symmetric triangular sets as well as trapezoidal ones.

The function is sent a pointer to the four values marking the limits of a nominally trapezoidal fuzzy set and the value for which the fuzzy membership of the set is to be found. The value is tested. If it is above or below the limits of the set its membership is zero. If it is between the two central values its membership is one as this is the top of the trapezium. Otherwise it must be on a slope of one of the sides and its membership is calculated by multiplying the distance it is into the set by the gradient of the side.

3.4.2 Activation - Navigation Controller

The fuzzy controller assumes the FAM matrix is fully populated but it is possible to deactivate a rule by giving it a higher rule number than the total number of defined output rules in which case it has no effect. The matrix style method is efficient for the dense rulesets used by this controller but is less efficient for small or incomplete rulesets. Because of the inherent symmetry in the navigation problem only half of the rules are stored in the matrix and those for symmetric positions about the goal inferred from these.

This function, named *activation3*, is passed a pointer to the empty output vector and a pointer to an array containing the three state variables. The set definitions and the rules are accessed via a pointer to the *fams* structure which is defined and initialised at the start of a test.

The function runs through the rules in the FAM matrix and adds the consequent of non-zero activations to the output vector. All entries in the FAM matrix are of the A AND B AND C implies D type where the fuzzy logic AND is a pairwise minimum. After this function the output fuzzy fit vector is filled with the relative activation of the FAM rules.

To avoid checking rules for which one of the sets is known to have zero activation and therefore zero output a record is kept on the first pass of where the first and last active sets occur. In subsequent passes with different first sets only those entries in the FAM matrix between the active sets are tested. Since sets which overlap are adjacent in the FAM matrix this search method significantly reduces the number of entries which need to be tested.

For the simulation program sections of code in this routine display which rules have been activated and their consequents on the screen. This has to be done inside the routine since this information is local to the function. On the vehicle this display routine is removed although it is possible to make the same information be passed to the message handler for downloading and inspection after the run.

Scanning the FAM Rule Matrix

The loop inside this procedure is the heart of the fuzzy control scheme and so is described in more detail. The loop operates on the *fams* structure which contains three arrays describing the trapezoidal fuzzy input sets and a three dimensional matrix which contains the FAM matrix linking the input fuzzy sets to output fuzzy sets. The structure also contains the values indicating how many fuzzy sets each input state variable has been split into. The FAM matrices used can be found in Chapter 4 as Tables 4.2 to 4.13.

The first loop runs through all sets of the first input state variable. If the membership of the first state variable in the current set is found to be non-zero then the next state variable is examined. On the first entry into this next loop it covers all the sets for the second state variable. On subsequent executions of the loop the rule matrix is only scanned between the first set with non-zero activation and the last set with non-zero activation. This is achieved

by storing the number of the first and last active sets found within the loop. The values defining the start and end values of the loop are then set for all remaining executions of the loop thus preventing unnecessary additional searches. The loop running through the possible sets for the third state variable operates in a similar manner to remove unnecessary searches. For the ruleset used with the controller this technique reduces the number of tests to be made from $4 \times 9 \times 15 = 540$ to $4 + 9 + 15$ tests during the first pass and up to 7 subsequent active rules giving a maximum of 35.

For each rule found to have non-zero activation the consequent output set is found from the rule matrix. If the set number for the third state variable is such that the consequent rule must be found by symmetry then the actual value is derived. The output fuzzy fit vector is updated by incrementing the appropriate output set activation by the minimum activation of the three input sets.

3.4.3 Activation - Velocity Controller

The activation of the rules governing velocity are found by the function *vactivation* which is passed a pointer to the empty output vector and a pointer to an array containing the state variables. The set definitions and the rules are accessed via a pointer to the *rules* structure which is a list of rules defined at the start of a test.

The activation of each rule is found by running through the list of statements associated with that rule which are the activations of particular input sets, as found by the routine *fuzzyval*. The inverse of a particular set can be signified by the setting of a flag in the list. This causes the fuzzy NOT to be applied, using one minus the activation of a set. The activations associated with a particular rule are AND'd together by finding the minimum value. This is then applied to the corresponding output vector for the rule if it exceeds the current activation. This gives the min-max inference method used by the velocity rules. The routine runs through the full list of rules to get the final activation of the fit vector.

3.5 Defuzzification Routines.

When the fuzzy fit vector has been filled a single output value needs to be found. This is done by a defuzzification scheme. These routines implement the defuzzification scheme

for the controller. They include definitions for simple 'centre of gravity' defuzzification and correlation product or correlation minimum association schemes. Also included are the routines which implement rule spreading and windowing and which combine the outputs of the inhibitive obstacle avoidance rules with the output of the navigation rules. This is covered in Section 5.7. The 'C' code for these routines can be found in the file `Defuzzy.c`.

3.5.1 Standard Defuzzification

The routine *defuzzy* is passed pointers to the output fuzzy fit vector and the output set definitions, a flag indicating the inference method to be used and the size of the fit vector. It returns the defuzzified output value. The output sets are stored in the same format as the input sets.

The function performs the sum:

$$\text{Output Centroid} = \frac{\sum_{i=1}^{n} c_i a_i}{\sum_{i=1}^{n} a_i} \tag{3.1}$$

where a_i = Activated area of the ith fuzzy output set.

c_i = Centroid of the ith fuzzy output set.

n = Number of entries in the fit vector

The activated area and the position of the centroid are calculated by the function *correlation* and are calculated differently depending on which method is used. This function is passed a set definition, the activation level of the set and the method to be used and returns the centroid and area of the activated set. A value of '1' for method causes correlation minimum to be used while a value of '2' selects the correlation product method. For the purposes of calculation the main difference between the two methods is that while correlation product simply shrinks the size of the whole set correlation minimum changes the position of the centre vertices. The area is calculated using the standard $\frac{1}{2}h(a+b)$ formula for the area of a trapezium. If the set is symmetrical then the centroid is simply the average of the two extreme points. Otherwise it is calculated by finding the area and centroid of the centre rectangle and side triangles of the set and combining them to give the centroid of the complete set.

3.5.2 Windowing and Rule Spreading Routines

The routines *apply_window, generate_bell, spread_activation* and *combine* apply the new techniques described in Section 5.7. The following are just descriptions of the way in which the routines operate.

The *apply_window* routine is passed a pointer to the fuzzy fit vector and a pointer to the array for the windowed version of the fuzzy fit vector. A value *max_winwidth*, indicating how far from the maximum activated rule the window is to be applied is also passed to the routine.

The centre point of the window is found by simply running through all the output rules to find the maximum. Ties between equally activated rules are resolved by an array of priority values which is set to give precedence to the larger steering angles and left turns. The routine starts with a window width, held in the variable *winwidth*, of 0 and then increments this, examining the new values in the fit vector this would add and stopping when it locates a value less than the threshold value (0.1) or it reaches the maximum window size.

Output set activation values which lie less than *winwidth* from the maximum value are copied to the output activation vector, others are set to zero.

The routine *generate_bell* fills an array *bellcurve* according to the following formula if $k \neq 0$

$$\text{bellcurve}\,[i] = e^{-k(i-r)^2} \tag{3.2}$$

and if $k = 0$

$$\text{bellcurve}\,[i] = \begin{cases} 1 & k = r \\ 0 & k \neq r \end{cases} \tag{3.3}$$

Where r is the number of existing output sets and k is the constant passed to the routine.

This curve is used by the routine *spread_activation* to spread the influence of the activated rules over the entire output universe of discourse. *Spread_activation* is passed pointers to the original output fuzzy fit vector, the required new vector and the *bellcurve* array.

The routine takes each entry in the fit vector and adds its value multiplied by the appropriate value on the normal distribution described by *bellcurve* to each set activation in the new output vector. Once this has been applied for all rules the maximum activation

is found. If this value exceeds one the activations are normalised to this value ensuring that the maximum activation of any output set is one.

The *combine* routine combines the output fuzzy fit vector produced by the navigation rules and the inhibitive mask vector produced by the obstacle avoidance scheme. Each entry in the fuzzy fit vector is multiplied by the corresponding value in the output mask. After this function the fuzzy fit vector contains the masked values.

3.6 Motion Simulation Routine

During operation the vehicle speed (v) and steering angle (ϕ) are updated at 100ms intervals. Various limits, as explained in Section 2.5, apply to the rates of change of the steering angle and the velocity and the simulation applies these in the same way as the motion control process on the vehicle does. For a given time interval Δt the steering angle and velocity are assumed to remain constant. This is true for the research vehicle during the 100ms between control updates but would only be an approximation for a vehicle steered and/or driven by continuously varying D.C. motors. A vehicle with velocity v will therefore travel a distance $d = v\Delta t$ on a fixed radius of curvature r which can be found from Equation 2.1 derived in the previous chapter. The change in orientation $\Delta\theta$ can then be found from Equation 3.4 which is simply a rearrangement of Equation 2.15.

$$\Delta\theta = d/r \tag{3.4}$$

There is clearly a special case where ϕ is zero, and $\Delta\theta$ is also zero which must be detected in order to avoid a division by zero error during calculation. Once the values of d, r and $\Delta\theta$ have been found the change in position can also be found by applying the previously derived equations 2.16 and 2.17.

The 'C' function which performs the calculations to simulate the motion of the vehicle can be found in the file `Agvsim2.c` and is named *new_pos*. This routine is passed a pointer to a structure which holds the current position, orientation, velocity and steering angle of the vehicle. It is also passed the demanded velocity and steering angle and the length of time for which these settings will be valid (the higher level sample interval of 200ms). The routine uses the steering angles to find the current and demanded values of the steering

constant K_{steer} by applying Equation 2.3 and then applies the limits to the change of velocity and steering constant discussed in Section 2.5. The routine applies the equations mentioned above to alter the values in the structure indicating the state of the vehicle using the duration of a low level control interrupt as Δt until the total time is reached when it returns the total distance travelled. If the set points are reached before this time Δt is set to the total time remaining to speed up the calculations. Since this routine executes much faster than the actual sample times on the vehicle many simulation tests can be done in the time taken to perform one 'real' run with the vehicle.

3.7 Information Display

At each step of the simulation, representing the duration of a control interrupt, information about the vehicle is displayed to the screen. An example of the visual output of the simulator can be seen in Figure 3.3. The left hand side of the screen displays the goal location, the current vehicle position and the occupancy grid information about the presence of obstacles. The right hand side of the screen is an information window which shows information about the current position of the vehicle and the workings of the controller. The position and orientation of the vehicle is shown in the top part of the information window. The orientation is indicated by `phi` rather than by the symbol θ used throughout this thesis for historical reasons. The current velocity and the value demanded by the velocity controller are also shown, followed by the values of the state variables. The labels used are `dist` indicating the distance to the goal, `d S` the heading error (θ_{HE}) and `d G` the goal error (θ_{GE}). Below this is shown the output steering angle from the controller and the present steering angle. The next set of information is about the internal state of the controller and is displayed from within the fuzzy logic functions since it is local to them. The activated rules are displayed in a shorthand format indicating the activated distance, θ_{HE} and θ_{GE} sets and the consequent output set. The output rules section shows the combined (windowed) fit vector and the mask vector so that the internal workings of the controllers are visible at each stage.

Figure 3.3: Visual Output from the Vehicle Simulator

Chapter 4

Fuzzy Navigation Controller

This chapter describes the development of a fuzzy logic controller to navigate an automated vehicle to a specified goal position and orientation. The controller was designed empirically using intuitive design principles and the reasoning behind the selection of the state variables, input sets and fuzzy rules are discussed. The full Fuzzy Associative Memory Banks[1], or rulesets, and the associated set definitions arrived at during different stages of the design process are presented. Changes made to the controller to improve its success and to allow for integration with the obstacle avoidance function are introduced and their effects illustrated through a series of tests. The effects of velocity on the performance of the rules are considered, particularly in respect of the limitations velocity places on steering and the results of tests at different, fixed, speeds are presented. A velocity controller is described and results are shown from some tests during which it was used. Finally some tests which include simulated position uncertainty are introduced to demonstrate how the controller copes with noise.

4.1 Control System Design

The fuzzy logic controller for the navigation system was designed by building up a framework from operator knowledge and then adapting the controller manually to fine tune the performance. It was a surprisingly simple task to design the rules for the controller since a

[1] FAM

sensible steering angle for any given position and orientation was easy to select intuitively.

Once the initial rules had been selected tests were performed in simulation and the rules modified as 'rogue' rules were identified from undesirable behaviour by the vehicle. Three distinct stages are presented here with the initial controller being known as the *Coarse* controller because it uses fewer input sets than the next design, the *Fine* controller. The third design presented here is the *Final* controller which represents the culmination of the navigation design process.

4.1.1 State Variable Selection

There are several possible candidates for the controller's state variables. Clearly two state variables are needed to represent the error in position of the vehicle relative to the goal and another to represent the error in orientation relative to the goal, giving three in total.

For the purposes of locating the vehicle within its environment (the laboratory) a cartesian coordinate system was used with its origin in a corner of the room and axes parallel to the walls. The orientation of the vehicle within this coordinate system θ was measured relative to the Y axis. The state variables must include reference to x_v and y_v the vehicle's current position, x_g and y_g the goal position, θ_v the vehicle's orientation and θ_g the desired orientation of the vehicle when it reaches the goal. An obvious pair of state variables are either a cartesian or polar set of coordinates relative to the goal position. With the origin position at the goal and an axis parallel to the goal orientation it becomes easy to visualise a desired path to the goal from a position in the coordinate system. The third state variable must involve the vehicle's orientation θ_v.

Figure 4.1 shows the state variables selected which are essentially for a polar coordinate system relative to the goal with the zero angle reference parallel to the goal orientation. The goal error θ_{GE} is therefore zero when the vehicle is directly behind the goal. For the third state variable there are two candidates. These are the heading error θ_{HE}, the difference between the heading θ_s direct to the goal and the vehicle's current heading or the orientation error θ_{OE} which is the difference between the vehicle's current heading and the goal heading. θ_{HE} was selected as values close to zero always indicate that the vehicle is headed for the goal position. This is in most cases the desirable state, particularly when the vehicle is a long way from the goal position or is in line with the goal orientation.

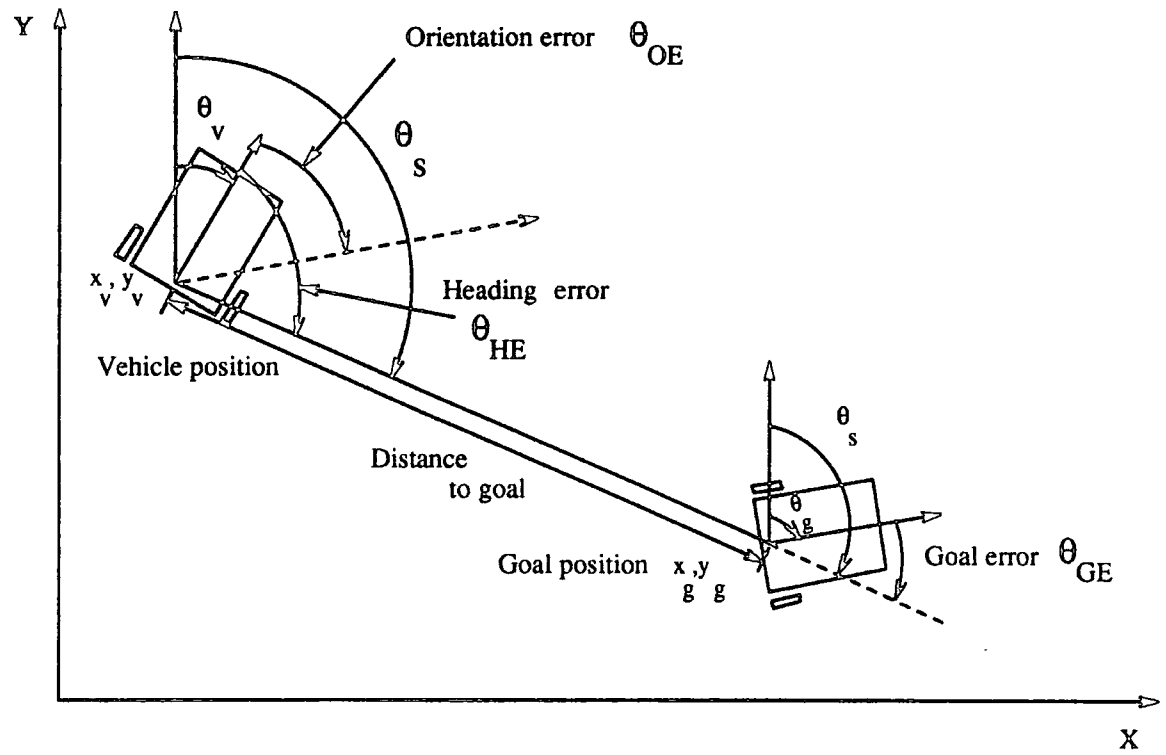

Figure 4.1: The State Variables in Relation to the Global Coordinate System.

There are some disadvantages however. Close to the goal the value of θ_{HE} can change by a considerable amount for a small change in the vehicle's position and in that position θ_{OE} is a better choice since it monitors the error in orientation independently of position and doesn't produce stability problems. For this reason θ_{OE} was used instead of θ_{HE} for the zero distance to the goal rules of the Final controller.

From Figure 4.1 it can be seen that the following four equations derive the values of the state variables from the global coordinates of the vehicle and its goal.

$$\text{Distance to goal} = \sqrt{(x_g - x_v)^2 + (y_g - y_v)^2} \tag{4.1}$$

$$\text{Goal error } \theta_{GE} = \theta_s - \theta_g \tag{4.2}$$

$$\text{Heading error } \theta_{HE} = \theta_v - \theta_s \tag{4.3}$$

$$\text{Orientation Error } \theta_{OE} = \theta_v - \theta_g \tag{4.4}$$

where

$$\theta_s = \arctan\left(\frac{x_g - x_v}{y_g - y_v}\right) \tag{4.5}$$

Set Label	Set Name	Set Label	Set Name
NA	Negative Away	PA	Positive Away
NB	Negative Big	PB	Positive Big
NP	Negative Perpendicular	PP	Positive Perpendicular
NL	Negative Large	PL	Positive Large
NM	Negative Medium	PM	Positive Medium
NS	Negative Small	PS	Positive Small
NZ	Negative Zero	PZ	Positive Zero
ZE	Zero	AW	Away

Table 4.1: Linguistic Names for the Goal and Heading Error Angle Sets

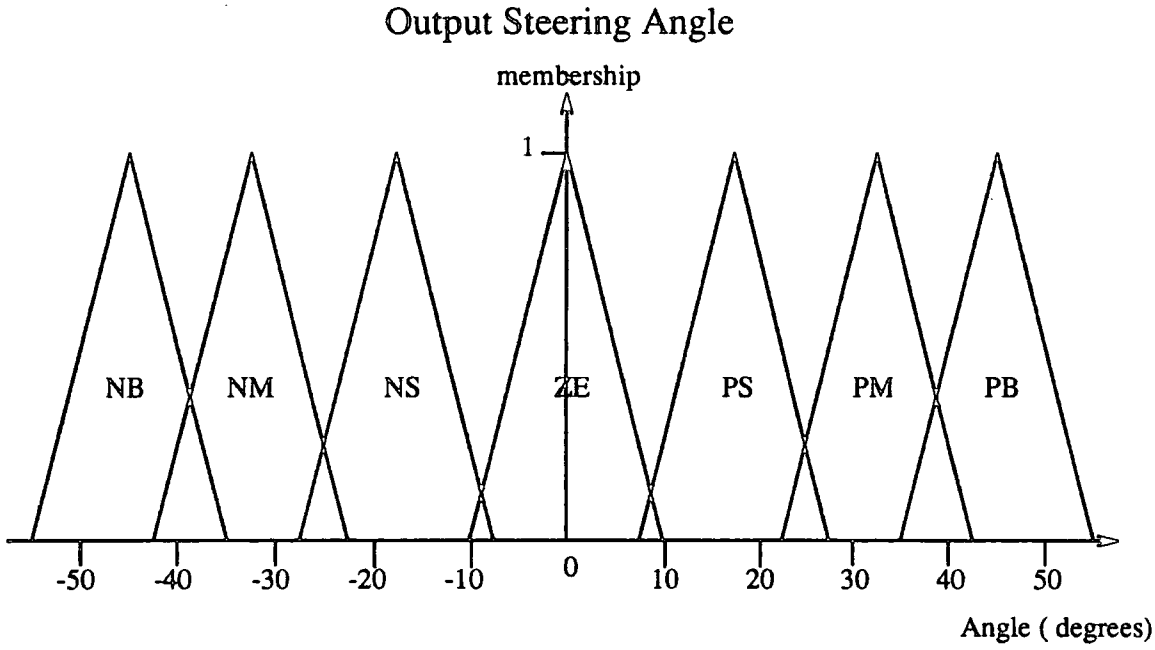

Figure 4.2: The Fuzzy Sets for the Output Steering Angle

4.1.2 Set Definitions

The definitions of the input and output sets were chosen to try and fit them to the positions where a change in behaviour was needed. Initially experience was gained by using a small number of sets, eight defined for both the ranges of θ_{HE} and θ_{GE} and four for the distance to the goal. These can be seen in Figures 4.3 and 4.6. The full names for the goal orientation error sets are given in Table 4.1 and where similar labels exist for the heading error the names are the same. The output sets were selected to give evenly spaced radii of curvature for each set and can be seen in Figure 4.2. The sets are the same for all three controllers and they are symmetric, triangular sets and so can be characterised by a centroid value and a weight for use with correlation product inference and centre of gravity defuzzification.

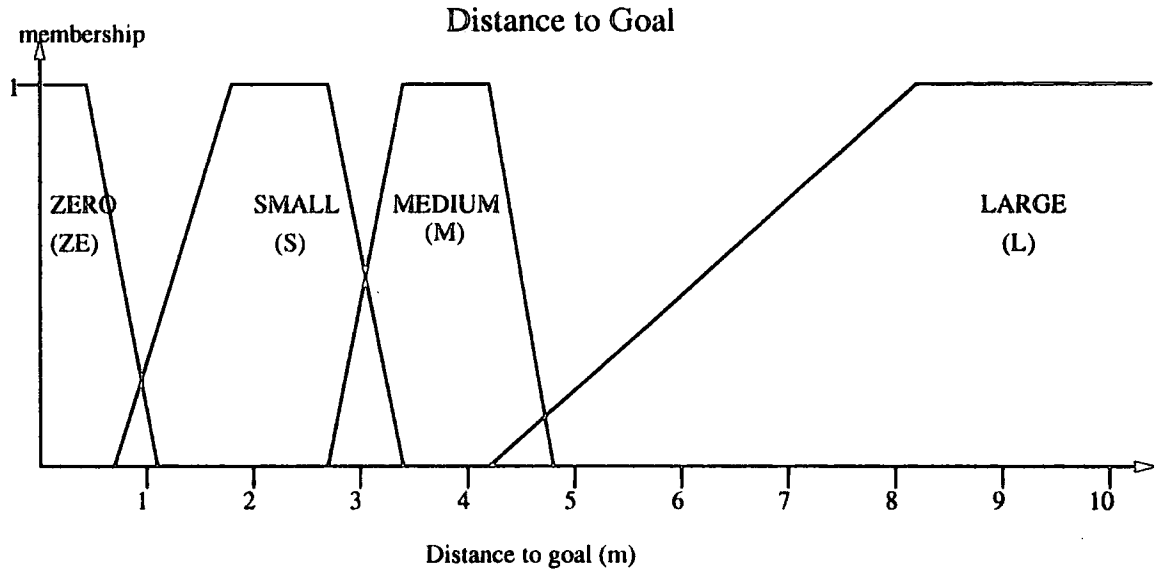

Figure 4.3: Coarse Controller, Fuzzy Sets for the Distance to the Goal

When the rules had been further developed and the performance of the first controller evaluated the θ_{GE} set definitions were increased to fifteen and grouped closer together around zero to allow for finer control when the vehicle was directly behind the goal position, as shown in Figure 4.7, these definitions were largely unchanged in the Final controller with only the zero heading error set being altered to be much narrower and so more representative of the final acceptable error. The full linguistic names for the distance sets are Large, Medium, Small and ZEro distances to the goal. The distance definitions were also brought closer to zero in the Fine controller and even closer still for the Final controller which has a very small zero distance set. This can be seen in Figures 4.4 and 4.5.

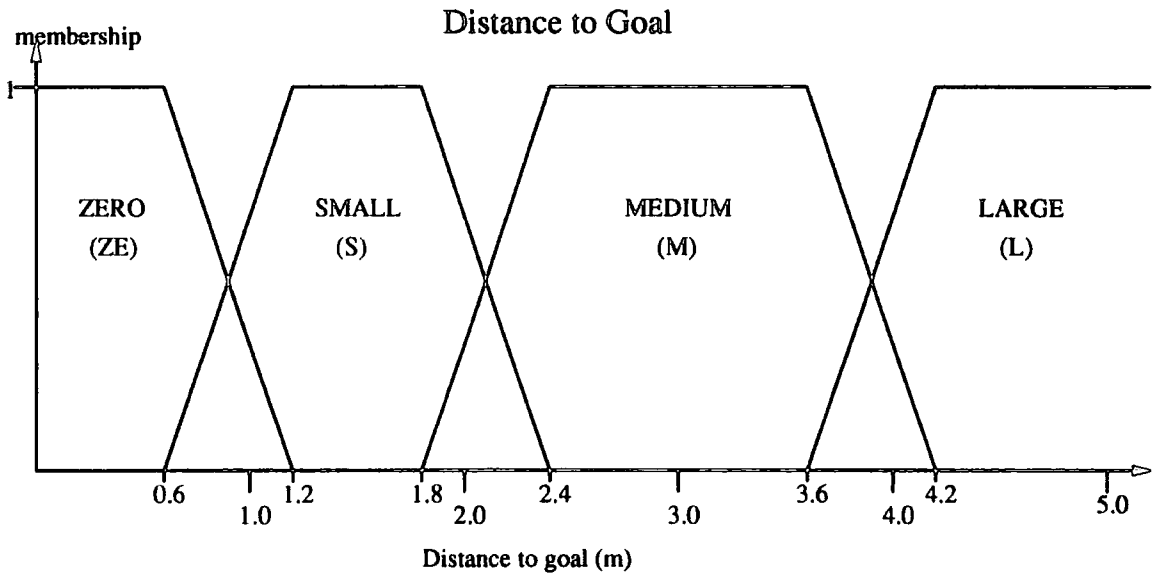

Figure 4.4: Fine Controller, Fuzzy Sets for the Distance to the Goal

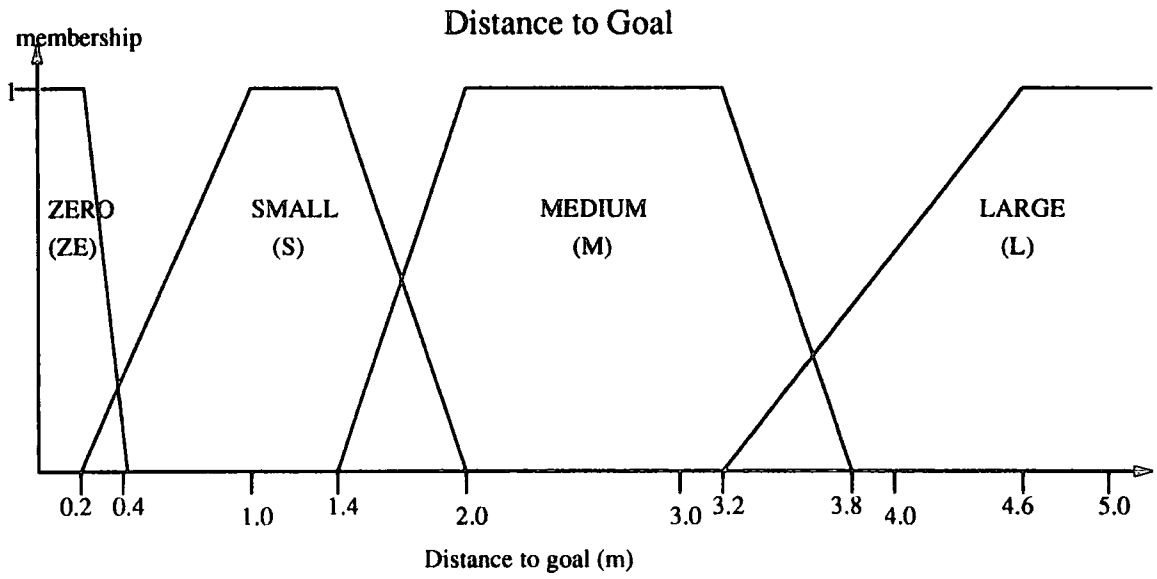

Figure 4.5: Final Controller, Fuzzy Sets for the Distance to the Goal

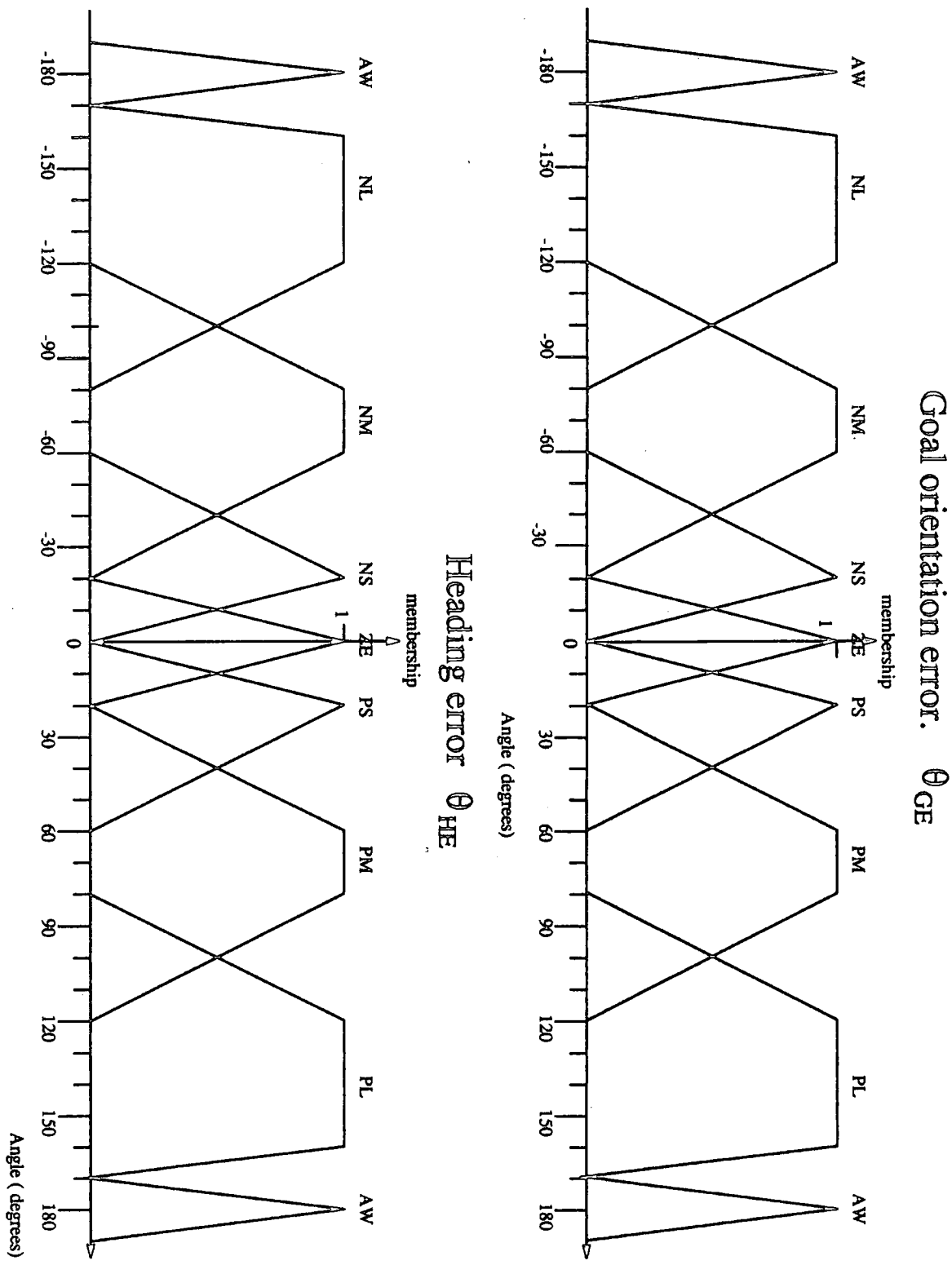

Figure 4.6: The Coarse Fuzzy Sets for the Goal and Heading Error Angles

Figure 4.7: Fuzzy Sets for the Goal and Heading Error Angles, Fine and Final Controllers

4.1.3 Rule Definition

To define the rules for the controller a set of principles or guidelines were used to determine for each possible combination of input sets how the vehicle should react. The principles were as follows.

1. If the vehicle is a long way from the goal position steer so as to head straight towards it, so θ_{HE} is steered towards zero.
2. If the vehicle is a medium distance from the goal attempt to move the vehicle to line up behind the goal heading towards it, so θ_{GE} approaches zero.
3. If the vehicle is a small distance from the goal position try to line up directly with the centre line, θ_{HE} and θ_{GE} should be zero.
4. If the vehicle is not orientated to succeed steer away from the goal for a new approach.
5. If the vehicle is almost on top of the goal position try to achieve the demanded final orientation, θ_{OE} or $\theta_{HE} + \theta_{GE}$ is zero.

On their own the tables convey little immediate information but they do provide the information necessary to duplicate the controller. The FAM bank matrix which stores all the rules for the controller is in these cases three dimensional as there are three state variables. Each complete matrix is therefore shown in four tables, one for each distance set. The rows are the heading error sets (θ_{HE}) and the columns are of the goal orientation error (θ_{GE}) except in the case of the zero distance rules for the Final controller where the rows represent the orientation error (θ_{OE}). For the Fine and Final controllers only the positive sets of goal orientation error are shown since the activated rules for the negative sets are derived by symmetry from these tables.

It is possible to see some of the ways in which the FAM banks relate to the principles described earlier and how the rules change between different controllers. The Large distance rules in the Coarse and Fine controllers can be seen to have a full line of zero steering angle outputs corresponding to zero heading error, as outlined in principle 1. In the Final controller however this has moved slightly so zero steering angles correspond to a small positive heading error which steers the vehicle into a better position for achieving a smooth approach to the final heading. The medium distance rules have to apply a balance between

principle 2 to approach the goal and principle 4 to steer away and attempt a realignment. It can be seen by looking at the bottom half of the Fine and Final rulesets that this is represented by the desired angle being changed from positive big to negative big. In the Fine ruleset this boundary is around the negative medium and negative perpendicular heading error sets. In the Final ruleset, however, this boundary has moved to be around the negative perpendicular or negative away sets as the error range over which the vehicle still tries to approach the goal has been increased. Significant differences can be seen between the zero distance rule banks. The direct affect of principle 5 can be seen in the Coarse rulebank where there are zero steering sets running diagonally across the ruleset to try and minimise $\theta_{HE} + \theta_{GE}$. In the Fine controller this has been altered to try and reach a compromise between limiting the orientation error and the position error. The Final controller rule bank for zero distance uses θ_{OE} instead of θ_{HE} which makes it easier to control the minimising of θ_{OE} directly. It can be seen from the ruleset that positive orientation errors induce a negative steering angle and vice versa. If the orientation error is close to zero and the goal error is large then the vehicle is steered towards the goal position slightly to further reduce the goal error.

θ_{GE}

θ_{HE}	PL	PM	PS	ZE	NS	NM	NL	AW
PL	NB	NB	NB	NB	NB	NB	NB	NB
PM	NB	NB	NB	NB	NB	NB	NB	NB
PS	NS	NS	NS	NS	NS	NS	NS	NS
ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
NS	PS	PS	PS	PS	PS	PS	PS	PS
NM	PM	PM	PM	PM	PM	PM	PM	PM
NL	PM	PM	PM	PM	PM	PM	PM	PM
AW	NB	NB	NB	NB	NB	NB	NB	NB

Table 4.2: Coarse Controller FAM Bank for a Large Distance to the Goal

θ_{GE}

θ_{HE}	PL	PM	PS	ZE	NS	NM	NL	AW
PL	NB	NM	NB	NB	PB	PB	PB	NB
PM	NM	PS	NM	NB	NB	NM	NB	NM
PS	PS	PM	PS	NS	NM	NB	NB	ZE
ZE	PM	PM	PM	ZE	NM	NM	NM	NM
NS	PB	PM	PM	PS	NS	NM	NS	ZE
NM	PB	PB	PB	PB	PM	ZE	PS	PM
NL	NB	NB	NB	PB	PB	PM	PB	PB
AW	NB	NB	NB	NB	PB	PB	PB	NB

Table 4.3: Coarse Controller FAM Bank for a Medium Distance to the Goal

θ_{GE}

θ_{HE}	PL	PM	PS	ZE	NS	NM	NL	AW
PL	NM	ZE	PS	ZE	ZE	PS	PB	NB
PM	NM	PS	NM	NB	NB	PM	ZE	NM
PS	PS	ZE	NS	NM	NM	PB	PS	PM
ZE	PM	NB	NS	ZE	PS	PB	NM	NM
NS	NS	NB	PM	PM	PS	ZE	NS	NM
NM	NS	NM	PB	PB	PM	NS	PS	PS
NL	NB	NS	ZE	ZE	NS	ZE	PM	PB
AW	NB	NM	PS	NM	NS	PM	PB	NB

Table 4.4: Coarse Controller FAM Bank for a Small Distance to the Goal

θ_{GE}

θ_{HE}	PL	PM	PS	ZE	NS	NM	NL	AW
PL	PB	PM	–	–	NS	NM	ZE	PM
PM	–	–	NB	NB	NM	ZE	–	–
PS	–	NB	NM	NS	ZE	PS	–	–
ZE	–	NB	NS	ZE	PS	PB	–	–
NS	–	NS	ZE	PS	PM	PB	–	–
NM	–	ZE	PM	PB	PB	–	–	–
NL	ZE	PM	PS	–	–	NM	NB	NM
AW	PM	–	PB	NS	NB	–	NM	ZE

Table 4.5: Coarse Controller FAM Bank for a Zero Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NB	NB	NB	NB	NB	NB	NB	NB
PP	NB	NB	NB	NB	NB	NB	NB	NB
PM	NM	NS	NS	NS	NS	NS	NS	NB
PS	NS	NS	NS	NS	NS	ZE	ZE	NS
ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
NS	PM	PM	PM	PM	PS	PS	PS	PS
NM	PB	PB	PB	PB	PB	PB	PB	PB
NP	PB	PB	PB	NB	NB	NB	NB	PB
NA	NB	NB	NB	NB	NB	NB	NB	PB

Table 4.6: Fine Controller FAM Bank for a Large Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NB	NB	NB	NB	NB	NB	NM	NM
PP	NM	NS	NS	ZE	NS	NB	NB	ZE
PM	ZE	PS	PS	PM	PS	ZE	NM	NB
PS	PS	PM	PB	PB	PB	PB	ZE	NS
ZE	PM	PB	PB	PB	PB	PB	PS	ZE
NS	PB	PB	PB	PB	NB	PB	PM	PS
NM	PB	PB	PB	NB	NB	NB	NB	PB
NP	NB	NB	NB	NB	NB	NB	NB	ZE
NA	NB	NB	NB	NB	NB	NB	NB	NM

Table 4.7: Fine Controller FAM Bank for a Medium Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NB	NM	NS	ZE	PS	NB	NM	NM
PP	ZE	ZE	ZE	PS	PB	NB	ZE	ZE
PM	PM	PB	PB	PB	NB	NM	NB	NB
PS	PB	PB	PB	PB	PS	ZE	NS	NS
ZE	NM	PB	NS	PB	PM	PB	PM	ZE
NS	NS	NS	NS	NB	PB	NB	PB	PS
NM	ZE	PS	ZE	NB	NB	NB	NB	PB
NP	PB	PB	PB	NB	NB	NM	NS	ZE
NA	PB	NB	NB	NB	NB	NS	PS	PM

Table 4.8: Fine Controller FAM Bank for a Small Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NS	ZE	PS	PB	NB	NM	NM	NB
PP	ZE	NB	NB	NB	NS	ZE	ZE	ZE
PM	PB	NS	NS	NB	PS	NM	NM	NB
PS	NB	ZE	ZE	NB	NM	NM	NS	NS
ZE	NB	PS	PS	NM	NM	NS	NS	ZE
NS	NB	PS	PB	NM	NS	ZE	PS	PS
NM	ZE	NB	NB	NS	ZE	PM	PB	PB
NP	NB	NM	ZE	NB	NB	NM	ZE	ZE
NA	ZE	PM	NB	NB	NB	ZE	PM	PB

Table 4.9: Fine Controller FAM Bank for a Zero Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NB	NB	NB	NB	NB	NB	NB	NB
PP	NB	NB	NB	NB	NB	NB	NB	NB
PM	NM	NS	NS	NS	NS	NS	NS	NB
PS	ZE	ZE	PS	ZE	ZE	ZE	ZE	NS
ZE	PS	PS	PS	PS	PS	PS	ZE	ZE
NS	PM	PM	PM	PM	PM	PS	PS	PS
NM	PB	PB	PB	PB	PB	PB	PB	PB
NP	PB	PB	PB	PB	PB	PB	PB	PB
NA	NB	NB	NB	PB	PB	NB	NB	PB

Table 4.10: Final Controller FAM Bank for a Large Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NB	NB	NB	NB	NB	NB	NM	NM
PP	NM	NS	NS	NS	NM	NB	NB	ZE
PM	ZE	PS	PS	PS	ZE	NS	NB	NB
PS	PS	PM	PB	PB	PB	PB	ZE	NS
ZE	PM	PB	PB	PB	PB	PB	PS	ZE
NS	PB	PB	PB	PB	PB	PB	PM	PS
NM	PB	PB	PB	PB	PB	PB	PB	PB
NP	NB	NB	NB	NB	PB	PB	PB	ZE
NA	NB	NB	NB	NB	NB	NB	NB	NM

Table 4.11: Final Controller FAM Bank for a Medium Distance to the Goal

θ_{HE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	NB	NM	NS	ZE	PS	NB	NM	NM
PP	ZE	ZE	ZE	PS	PB	NB	ZE	ZE
PM	PM	PB	PB	PB	NM	NM	NM	NB
PS	PB	PB	PB	NS	PS	NS	NM	NM
ZE	NM	PB	NS	PB	PM	PM	PS	ZE
NS	NS	NS	NS	PB	PB	PB	PB	PS
NM	ZE	PS	ZE	NB	NB	NB	PB	PB
NP	PB	PB	PB	NB	NB	NM	NS	ZE
NA	PB	NB	NB	NB	NB	NS	PS	PM

Table 4.12: Final Controller FAM Bank for a Small Distance to the Goal

θ_{OE} \ θ_{GE}	PA	PB	PP	PL	PM	PS	PZ	ZE
PA	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE
PP	ZE	ZE	ZE	NB	NB	NB	NB	NB
PM	NB	NB	NB	NB	NB	NB	NB	NB
PS	NS	NS	NS	NS	ZE	NM	NB	NS
ZE	PS	PM	PM	PM	PS	PS	ZE	ZE
NS	PS	PS	PS	PM	PM	PM	PS	PS
NM	PB	PB	PB	PB	PB	PB	PB	PB
NP	ZE	ZE	ZE	ZE	ZE	ZE	ZE	PB
NA	ZE	ZE	ZE	ZE	ZE	ZE	ZE	ZE

Table 4.13: Final Controller FAM Bank for a Zero Distance to the Goal

The following examples illustrate how to use these tables, using the Final controller sets and rules. If the vehicle is a large distance, such as 5m, from the goal, facing directly towards the goal position but is 90° out on the desired final orientation then from Figure 4.5 the distance to the goal has membership 1.0 in the Large distance set. Considering the heading error sets in Figure 4.7 the heading error has membership 1.0 in the zero (ZE) set and the goal error a membership of 1.0 in the positive perpendicular (PP) set. The selected output set from the FAM bank would therefore be found in the Large distance matrix on the row labeled ZE and the column headed PP which turns out to be PS. This indicates the vehicle should be turning slightly right so it can get into position to approach the goal from behind.

In general there will be more than one rule active at any time, depending on the number of overlapping sets. Consider the situation shown in Figure 4.8. The vehicle is 1.55m from the goal position. It has a heading error of $-15°$ and a goal error of 15°. From the distance sets, Figure 4.5, it can be seen that 1.55m is a member of two sets, Medium to degree 0.25 and Small to degree 0.75. From Figure 4.7 it can be seen that the heading error of $-15°$ is a member of just the set NS to degree one while the goal error is a member of two sets PS and PZ to degree .5 in each one. This set of values will therefore activate four rules, these being:-

M,NS,PS $\mapsto$ PB
M,NS,PZ $\mapsto$ PM
S,NS,PS $\mapsto$ PB
S,NS,PZ $\mapsto$ PB

The first rule is activated to the minimum of Medium (0.25), Negative Small heading error (1.0) and Positive Small Goal error or min(0.25, 1.0, 0.5) which is 0.25. By the same method the other rules have overall activations of 0.25, 0.5 and 0.5 respectively. This gives a summed activation level of 0.25 PM and 1.25 PB [2]. Using the correlation product method this defuzzifies to $(0.25 \times 33°) + (1.25 \times 45°) \div (0.25 + 1.25) = 43°$.

[2] In the control software these output values would be normalised giving activations of 0.2 and 1.0 which defuzzify to the same value.

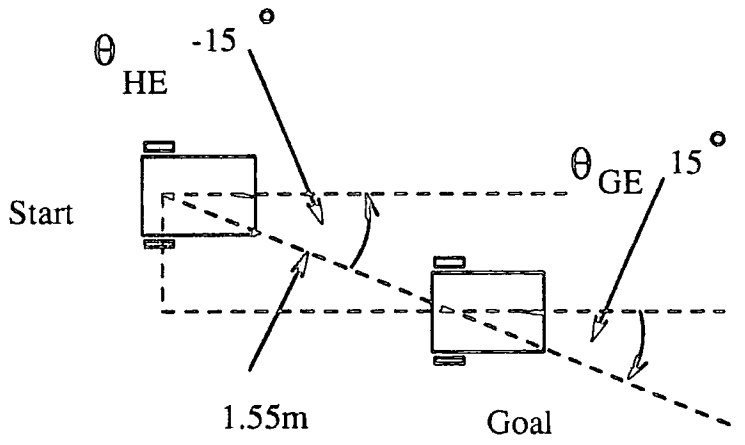

Figure 4.8: Vehicle Position for Set Activation Example

4.2 Initial Problems

The fuzzy logic controller described in the previous sections is capable of driving the simulated vehicle to the goal position and orientation from any arbitrary position. Depending on the initial conditions and the allowed error in the final position the vehicle may require a second attempt at reaching the goal. This is allowed for in the controller which was designed to move the vehicle around from a narrow failure to make the attempt again. During the design process some problems were observed which caused the controller to fail in certain situations.

4.2.1 Conflicting Rules

The first set of problems were caused by the defuzzification method and involve the controller being unable to decide between two contradictory rules. This can occur for instance if the vehicle is facing away from the goal precisely on the angle reference so that there is no way of making a choice between turning left or right, the goal is just as far in either case. For a fuzzy logic controller the result is that the activation of the Negative Big steering angle set is exactly the same as the activation of the Positive Big steering angle set with the result that the defuzzified value is zero. The effect of this is that, unable to decide between steering right or left, the vehicle continues away from the goal forever.

Kosko [39] notes this problem when evaluating the truck backer example in his book. He suggests that a rule could be introduced specifically to cope with this situation which essentially would push the defuzzified output away from zero by adding activation to a

non zero set. A second solution would be to adopt a maximum-membership defuzzification scheme which involves choosing the most activated set rather than averaging their influence as in the centre of gravity method. This method, as Kosko points out, has the disadvantage of discarding all the information held in the other activated sets and in a practical sense reduces the number of possible controller outputs to the number of output sets chosen.

In general it is possible to design around this problem but it is true that in certain situations the centre of gravity method does not give the desired result. Even if the problem is not as acute as to cause a complete failure the fact that conflicting rules will pull the output towards the global mean of all the sets, in this case zero, reduces the effectiveness of the controller.

4.2.2 Failure Caused by Dynamic Equilibrium

Although the design of the controller with a constant forward speed ensures that the controller will never fail in static equilibrium and be unable to move it is possible for the vehicle to become locked in a closed path which never reaches the goal. In particular an early problem was associated with approaching the goal at too steep an angle. As the vehicle got close the rules to steer the vehicle around in a loop for another attempt were activated as the angle was too steep and the vehicle turned around. The steering angle associated with this was too small however and resulted in the approach being too steep from the opposite side and the vehicle became locked in a figure of eight path just in front of or around the goal. An example of this can be seen in Figure 4.9 with the Coarse ruleset where the vehicle has adopted a path which always fails to reach the goal. This problem can be removed by careful adjustment of the rules but could be reintroduced when the effect of obstacles is taken into account. To remove this problem it would be advantageous to have some form of supervisory system higher in the control hierarchy to watch the vehicle's motion and correct such behaviour.

4.2.3 Obstacle Avoidance Considerations

An ideal obstacle avoidance scheme would not affect the performance of the navigation controller unless it was essential to avoid an obstacle. In practice however, as will be described in the next chapter, there is some effect and additionally possible course changes

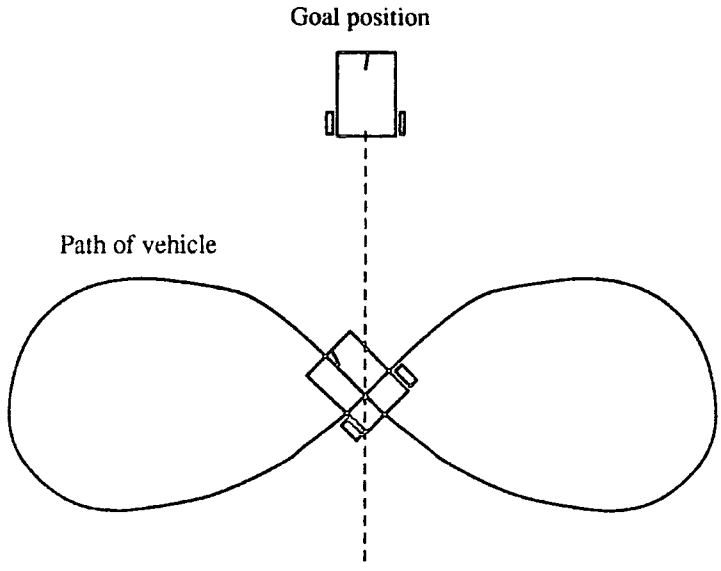

Figure 4.9: Example of Dynamic Equilibrium in a Faulty Controller Design

made to avoid an obstacle need to be allowed for by the controller. In particular it was found to be important to design the controller to cope with steep approaches to the goal rather than just trying to line up directly behind the goal a reasonable distance from it in order to cope with situations where the direct approach is blocked.

In particular modifications to the defuzzification method were made which are called windowing and rule spreading and are introduced in depth in Section 5.7. In Section 5.9 these modifications are shown to alleviate the conflicting rule problem described earlier in this section and are essential to allow the obstacle avoidance method to be combined with the navigation controller. In order to allow for the effect of these changes on the rules the navigation design process was carried out with these techniques in operation.

The inclusion of these techniques modifies the controller structure to the form shown in Figure 4.10. The state variables are fuzzified so they are represented by degrees of membership of fuzzy sets. The rules stored in the FAM bank correlate these input set membership values to activation values for output sets. This results in a fuzzy fit vector containing values representing the degree to which each possible output steering angle set would achieve the desired path to the goal. This fuzzy fit vector then has the activation values spread by the rule spreading technique and is windowed before defuzzification is applied. The use of these techniques does not significantly alter the performance of the navigation controller except to alleviate conflicts within the ruleset but, as will be shown in the next chapter, forms an essential part of the obstacle avoidance scheme.

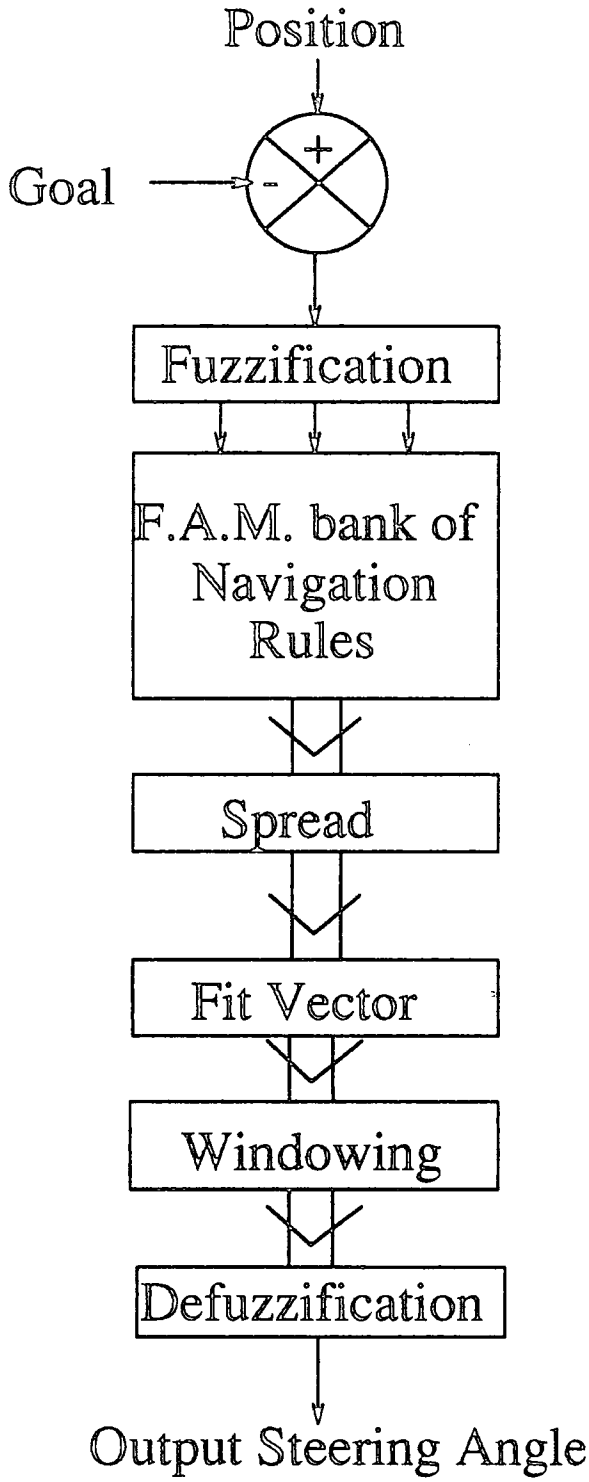

Figure 4.10: Controller Block Diagram, including Windowing and Rule Spreading

4.3 Performance Testing of the Navigation Rulesets

In order to refine the rules once the initial ruleset had been designed a set of tests were devised to enable the performance of the controller to be evaluated and the effect of different rulesets compared. All the tests operated with a goal position at the coordinates 7.0m, 7.0m and the goal orientation as zero degrees, that is, aligned with the Y axis. The first tests were designed to enable the performance of the navigation controller to be compared and improved and a second set were designed to see how well the controller could cope with approaching the goal from positions close to it but with large positional or angular errors which it might be forced to adopt by avoiding obstacles close to the goal. All tests were carried out with a sample time of 200ms and at a fixed velocity of 0.1m/s.

4.3.1 Test 1

The first test used involves a starting position three metres behind and one metre to the left of the goal with the vehicle facing 90° from the desired final orientation. In this case all that is required is for the vehicle to undertake a left turn to line up behind the goal and then proceed towards it.

In order to show how the imposition of the windowing and rule spreading techniques affect the controllers results are included for this first test both with and without windowing. The path taken by the vehicle can be seen in Figures 4.11 and 4.13 while the reduction of the angular error as the vehicle approaches the goal can be seen in Figures 4.12 and 4.14. The test performed without windowing or rule spreading shows that the Coarse rules take a smooth path to the goal approaching it in an exponential decay-like manner from the left hand side. The finer rules overshoot the goal before curving back in and the Final ruleset has a slight overshoot but takes a shorter path to the goal. With the windowing and rule spreading active the performance of the Coarse controller has been degraded so it overshoots the goal. The finer ruleset corrects this fault a little but still overshoots the goal. Using the Final rules, however, larger steering angle sets have been selected to prevent and correct the overshoot and the vehicle achieves the goal position with only a small error in final orientation.

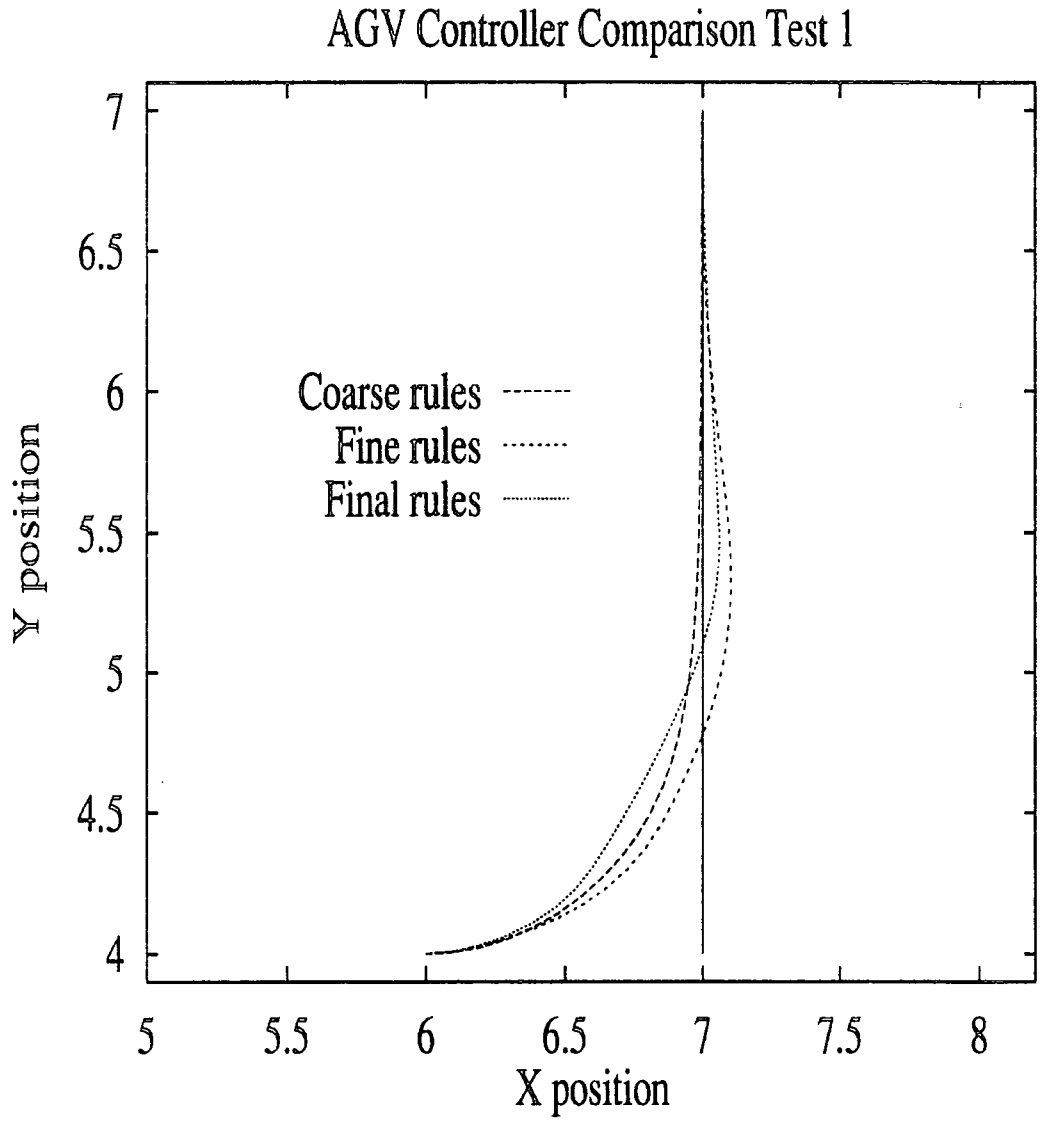

Figure 4.11: Path for Test 1, without Windowing and Rule Spreading.

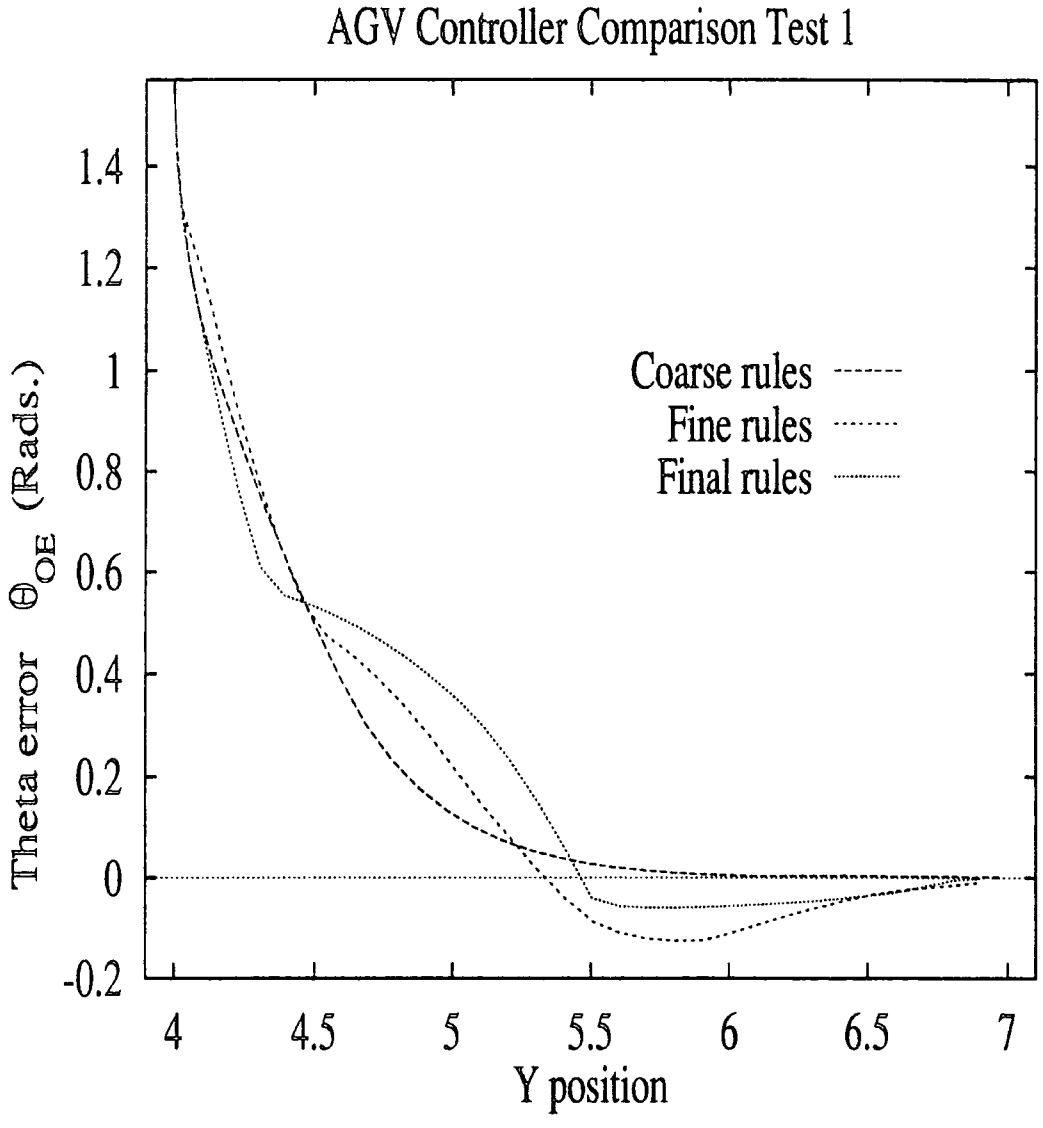

Figure 4.12: Angle Error During Test 1, without Windowing and Rule Spreading.

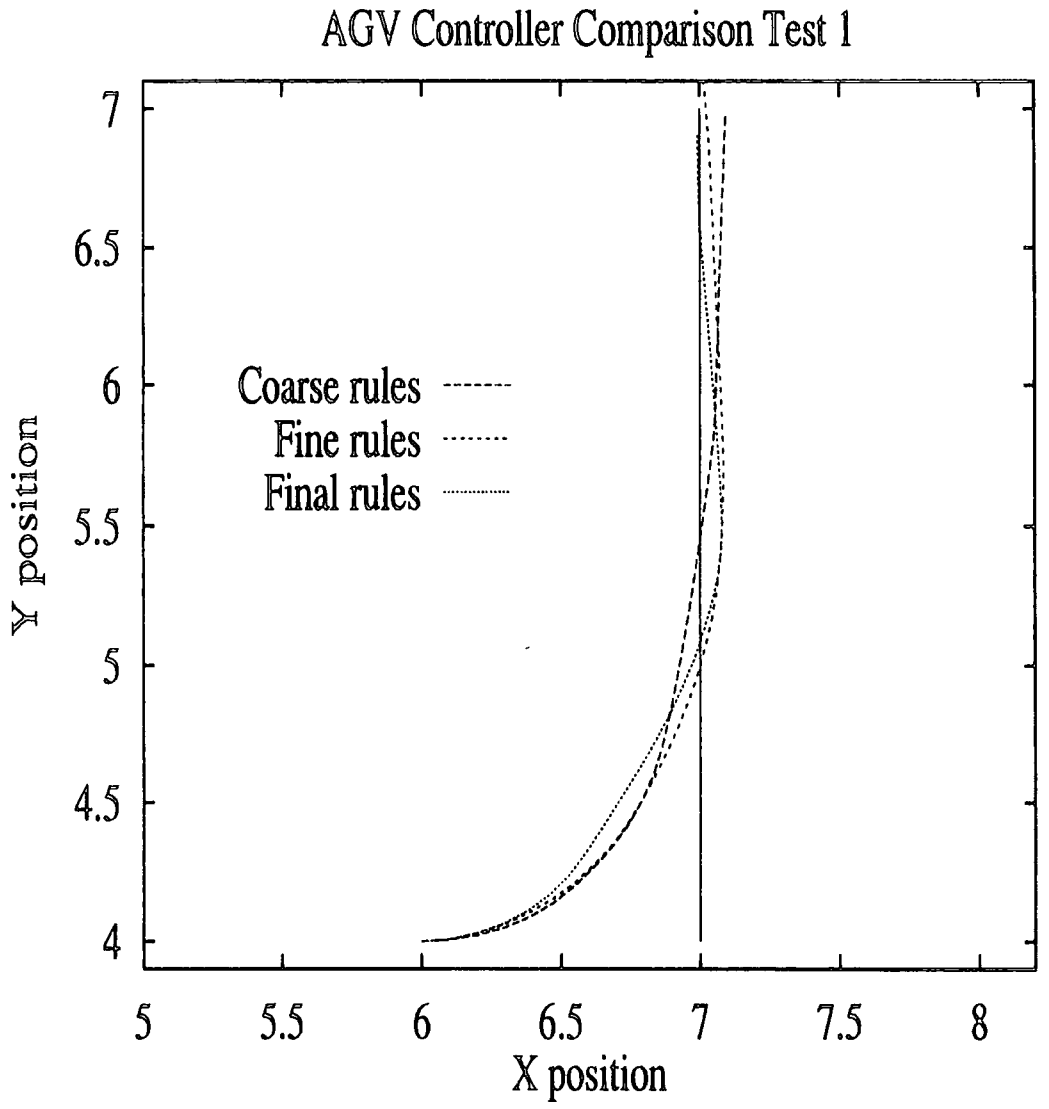

Figure 4.13: Path for Test 1, using Windowing and Rule Spreading.

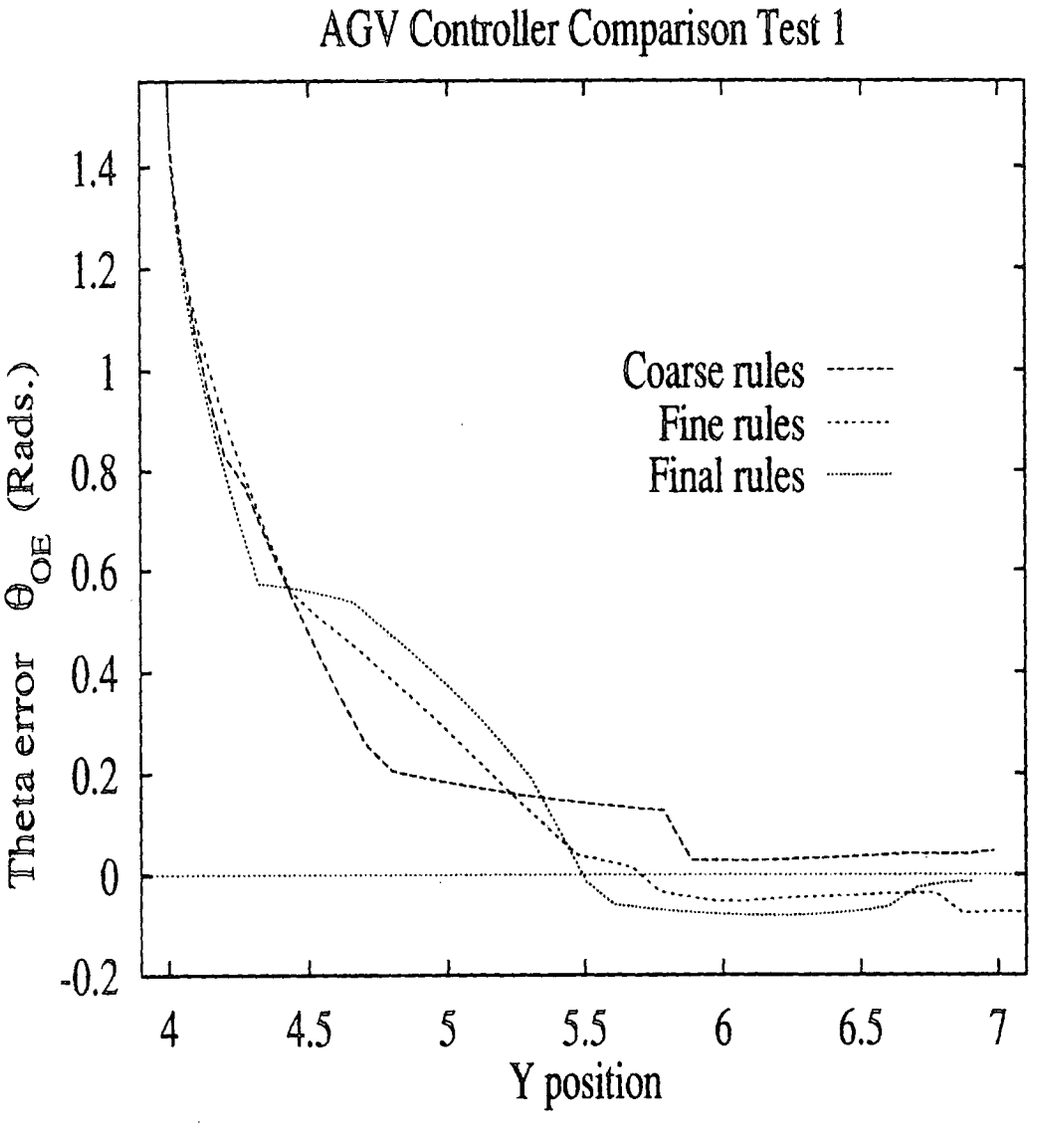

Figure 4.14: Angle Error During Test 1, using Windowing and Rule Spreading.

4.3.2 Test 2

The second test involves the same starting point but the vehicle is initially aligned with the goal orientation thus requiring an S shaped path to reach the goal location with the required alignment. Figure 4.15 shows the paths taken to the goal and it can easily be seen that the early, Coarse, ruleset fails to line up with the goal at all and passes the goal position well to the left hand side. The finer ruleset produces a much better response which is improved upon in the Final rules where the rules controlling the alignment with the extended centerline of the goal produce larger steering corrections and the vehicle achieves the final position. The graph of the alignment error in Figure 4.16 shows the oscillatory problems in the Coarse ruleset driven by rules trying to force alignment with the goal orientation acting too early and interacting badly with the rules trying to reduce the X error. The adjusted rulesets show a much better performance with the error in angle smoothly increasing as the vehicle is turned to approach the extended centerline of the goal and then reduced to zero, with a slight overshoot.

4.3.3 Test 3

The third test was designed to examine how well a controller coped with an approach with a larger initial error in X position. The starting orientation is the same as the previous test but the vehicle starts two metres further to the left giving 3m error in X and Y. This gives rise to a more extended S shaped path where the initial aim is to get close to the extended centreline of the goal position while being far enough from the goal to correct the orientation before attaining it. The paths taken to the goal are displayed in Figure 4.17 and the alignment error in Figure 4.18. Once again the Coarse ruleset fails to achieve the goal but passes close, this time to the right. The initial set of finer control rules overshoot the centerline by too much and fail to correct the trajectory of the vehicle. In the Final ruleset the overshoot is only slight and the vehicle achieves the final position with a small angular error.

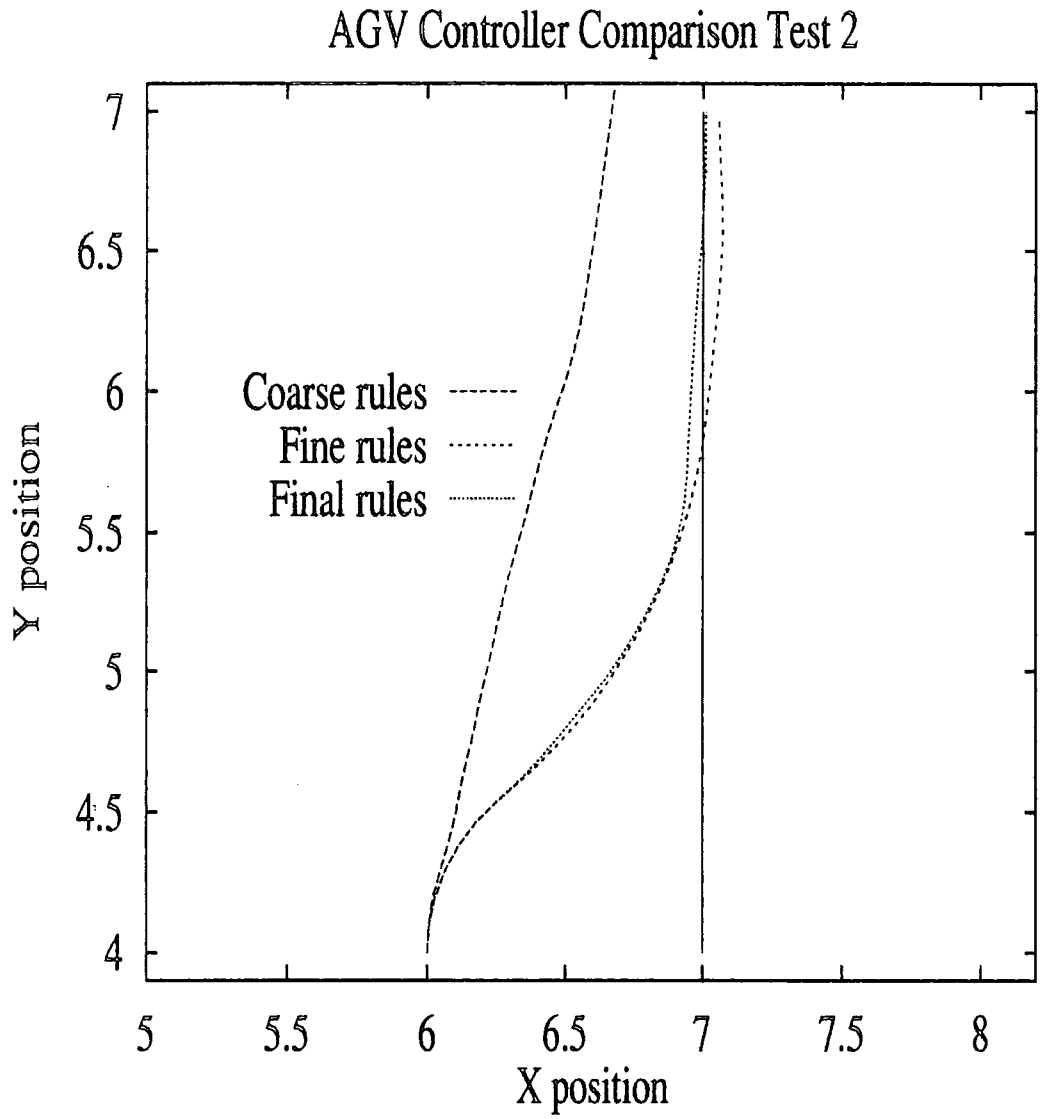

Figure 4.15: Path for Test 2.

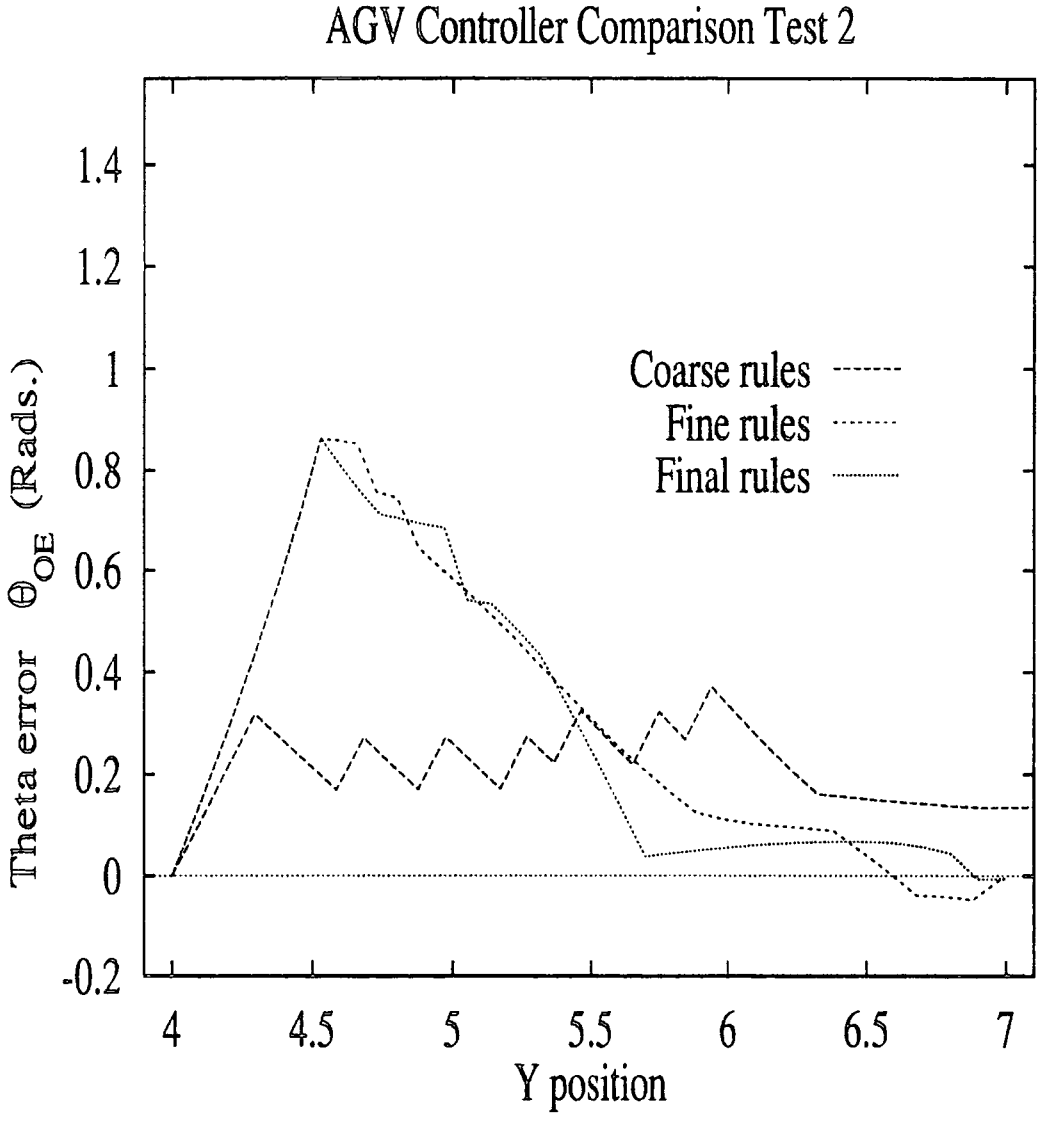

Figure 4.16: Angle Error During Test 2.

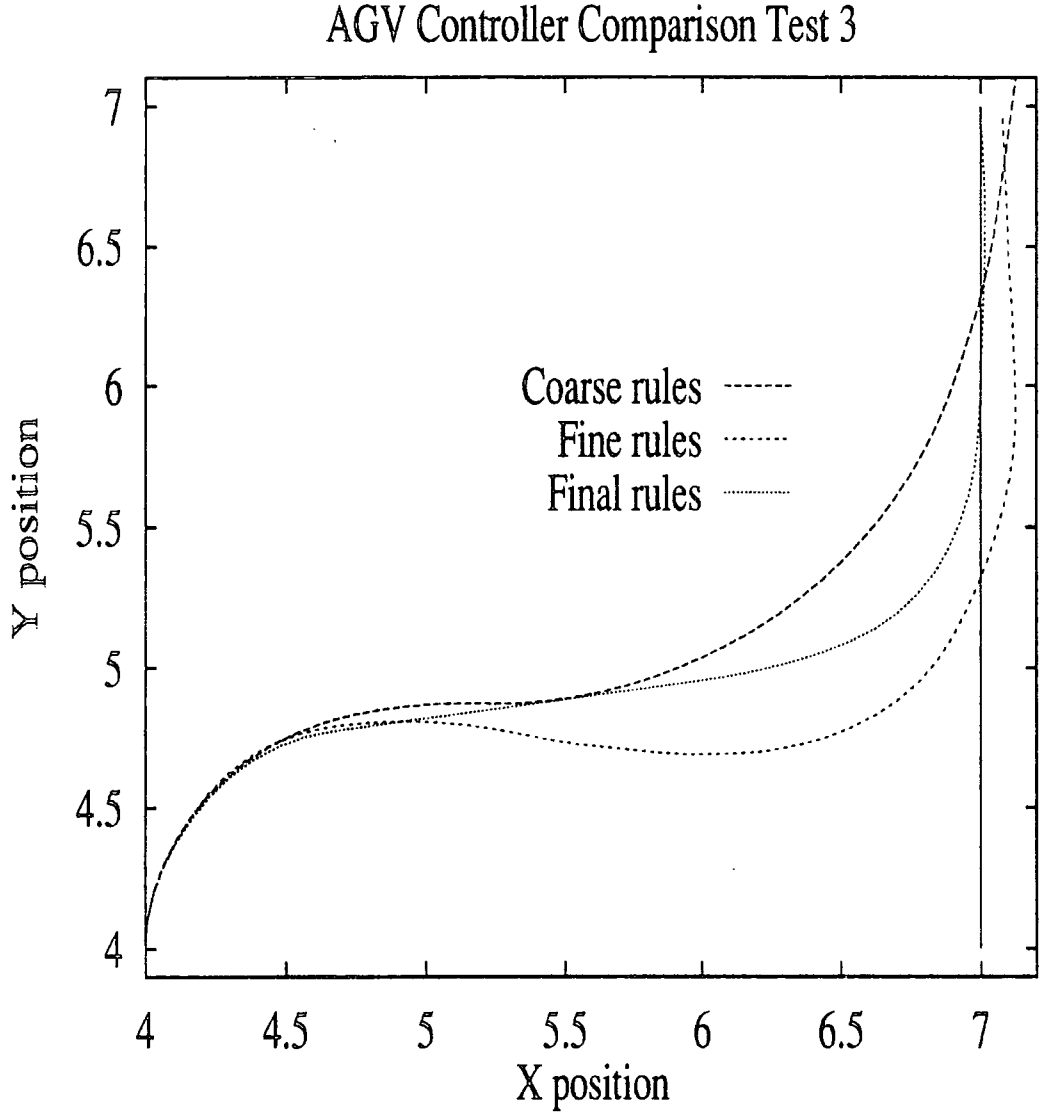

Figure 4.17: Path for Test 3.

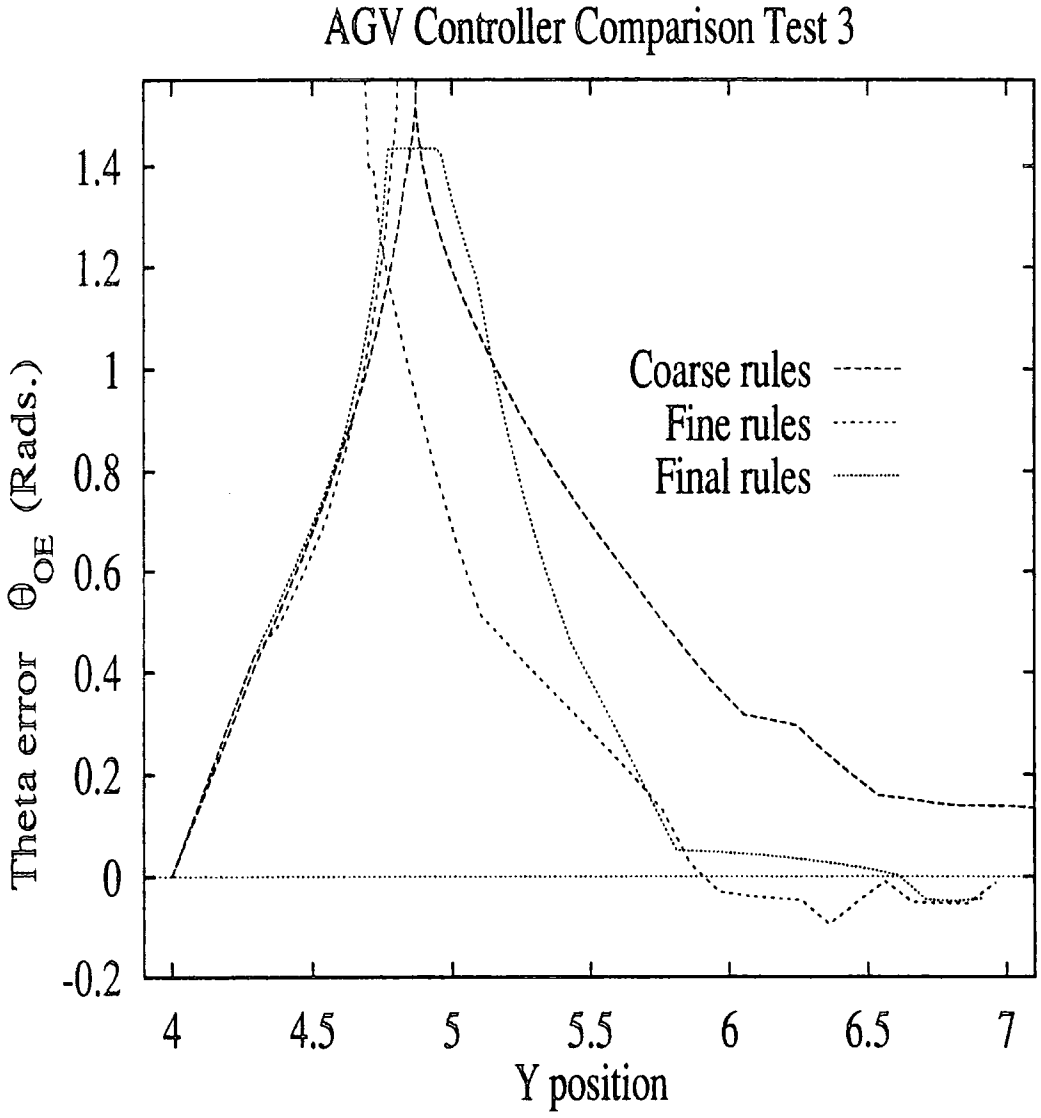

Figure 4.18: Angle Error During Test 3.

4.3.4 Test 4

The fourth test has the vehicle starting in a fairly unfavourable position, that is three metres to the left of the goal facing in the opposite direction to the goal orientation. A semi-circular or U shaped trajectory is therefore required to attain the goal position and orientation. As shown by Figures 4.19 and 4.20 the design process has improved the performance from a wide loop which fails to finally line up at the goal position produced by the Coarse ruleset to the Final ruleset's performance which lines the vehicle up with the centre line using a tighter loop than the others and achieves the goal position.

4.3.5 Final Controller Docking Performance Tests

The performance of the Final controller close to the goal was tested and adjusted to try and give successful docking performance when the vehicle only had a short distance in which to make corrections. In tests 5 and 6 the vehicle was started one metre and one point five metres directly behind the goal and tests run to discover the maximum angle away from the goal orientation it could start at and still reach the goal successfully in the first pass. These angles were 39° and 58° respectively and the paths taken can be seen in Figure 4.21. In the other tests the vehicle was started similar distances back from the goal and with the start orientation equal to the goal orientation. Repeated tests were made to discover the maximum perpendicular displacement from the centre line which could be applied before the vehicle failed to achieve the goal in the first pass. This is also shown in Figure 4.21 and the maximum displacements are 0.2m when the vehicle is 1.0m behind the goal and 0.7m when the vehicle is 1.5m behind the goal.

The values used in these tests were increased as much as was possible by adjusting the ruleset in order to allow the controller to cope with large positional or angular errors close to the goal position which although not normally produced if the controller had a long clear run to the goal could be caused when avoiding an obstacle.

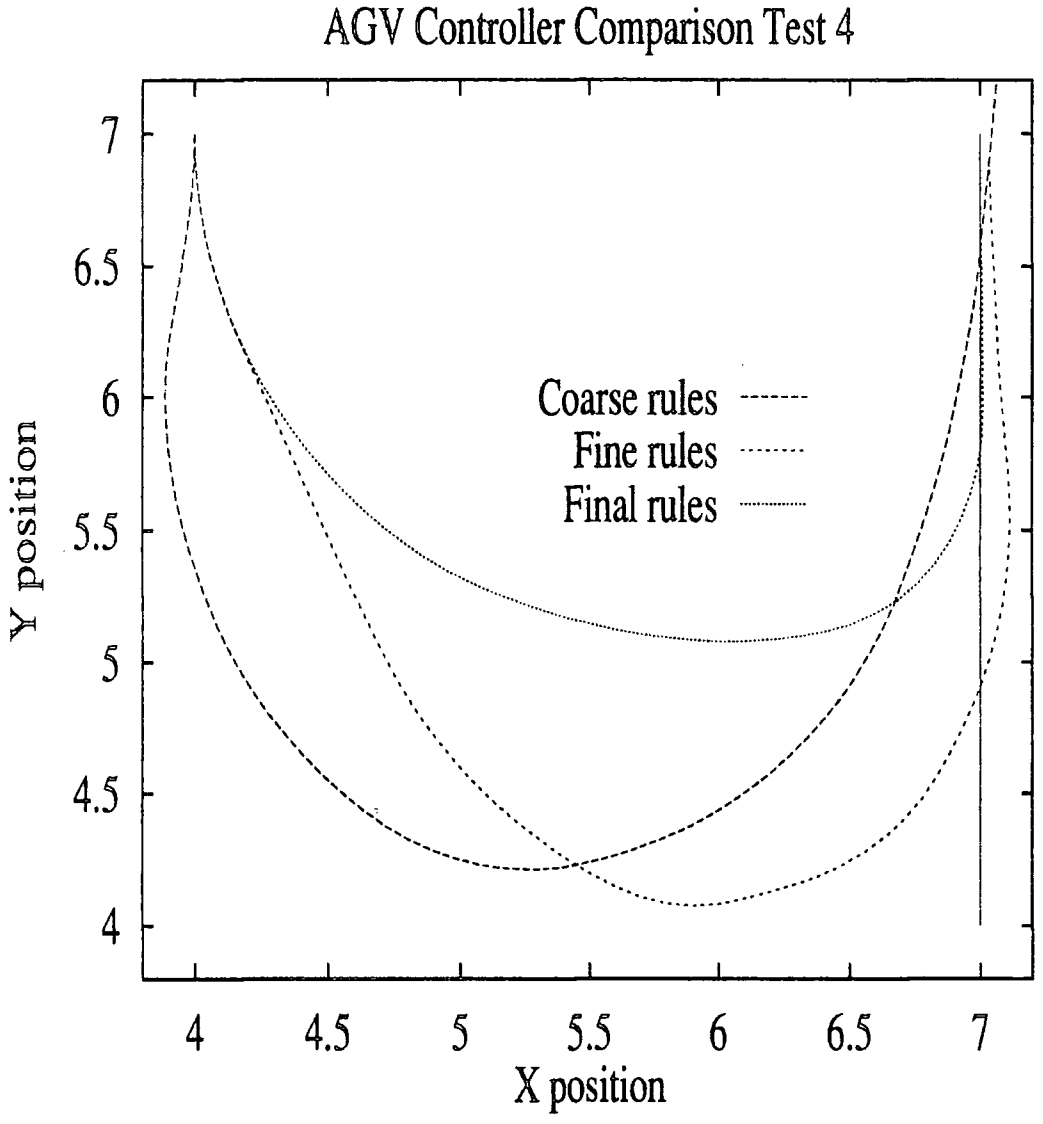

Figure 4.19: Path for Test 4.

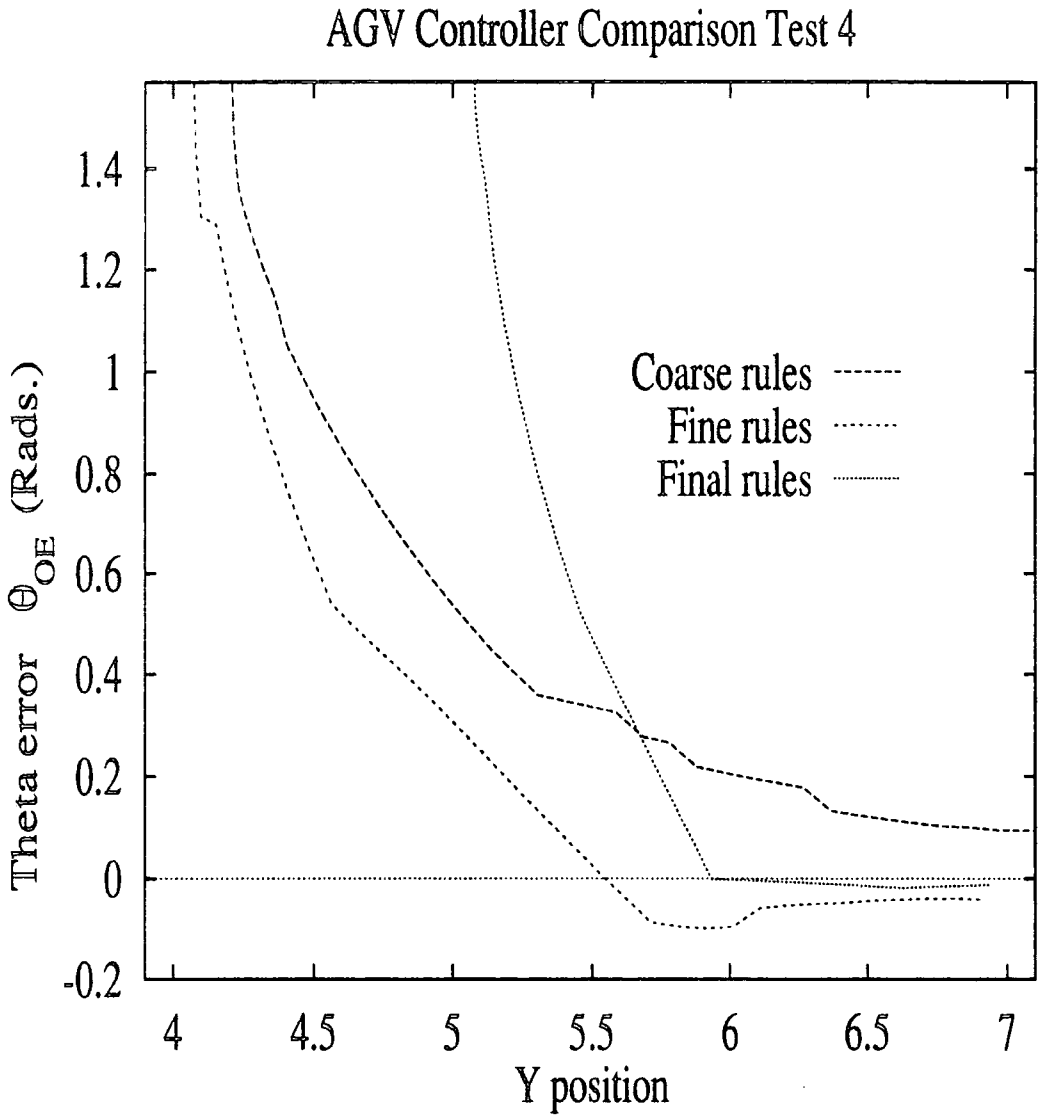

Figure 4.20: Angle Error During Test 4.

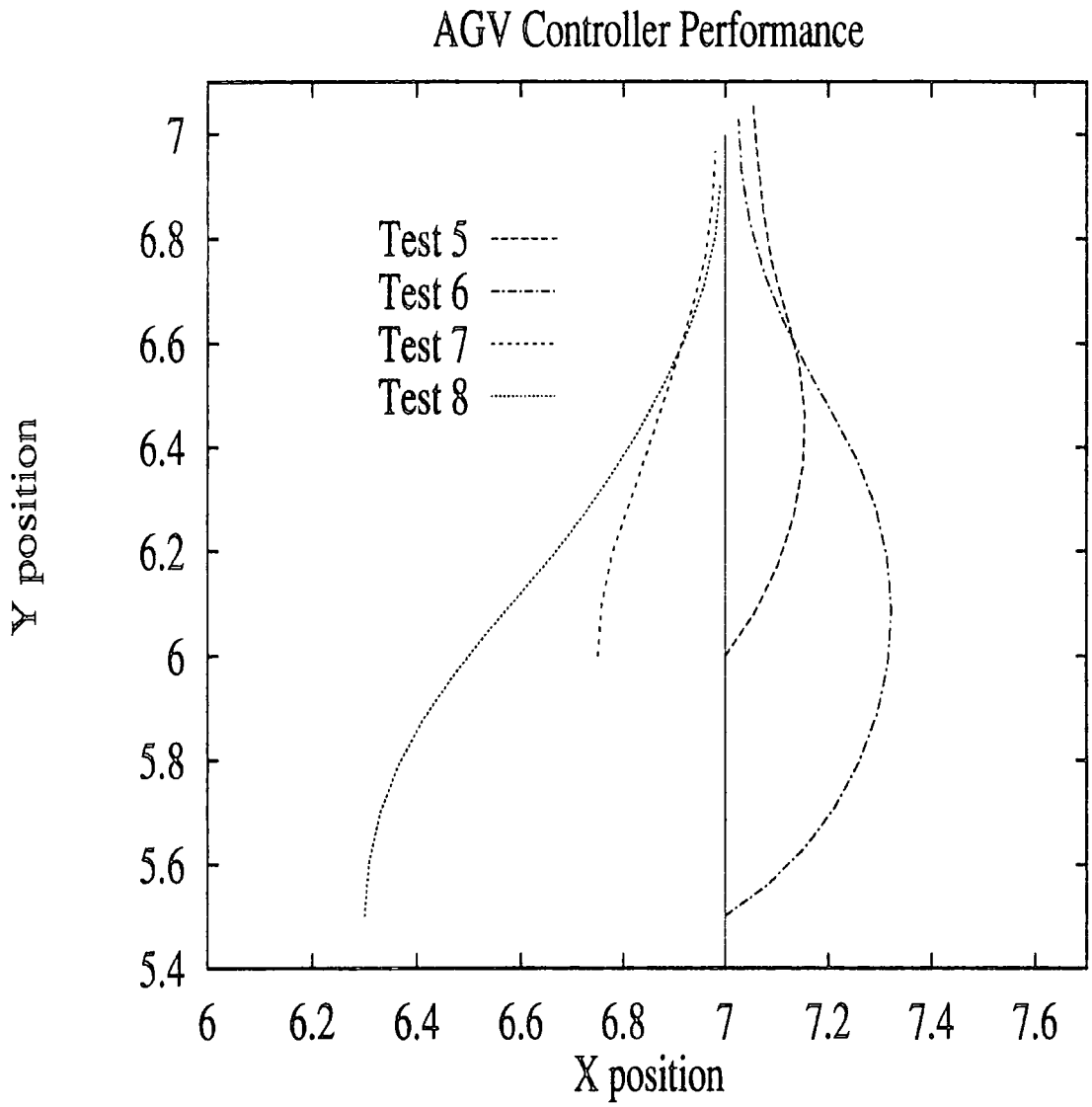

Figure 4.21: Paths for Tests 5-8.

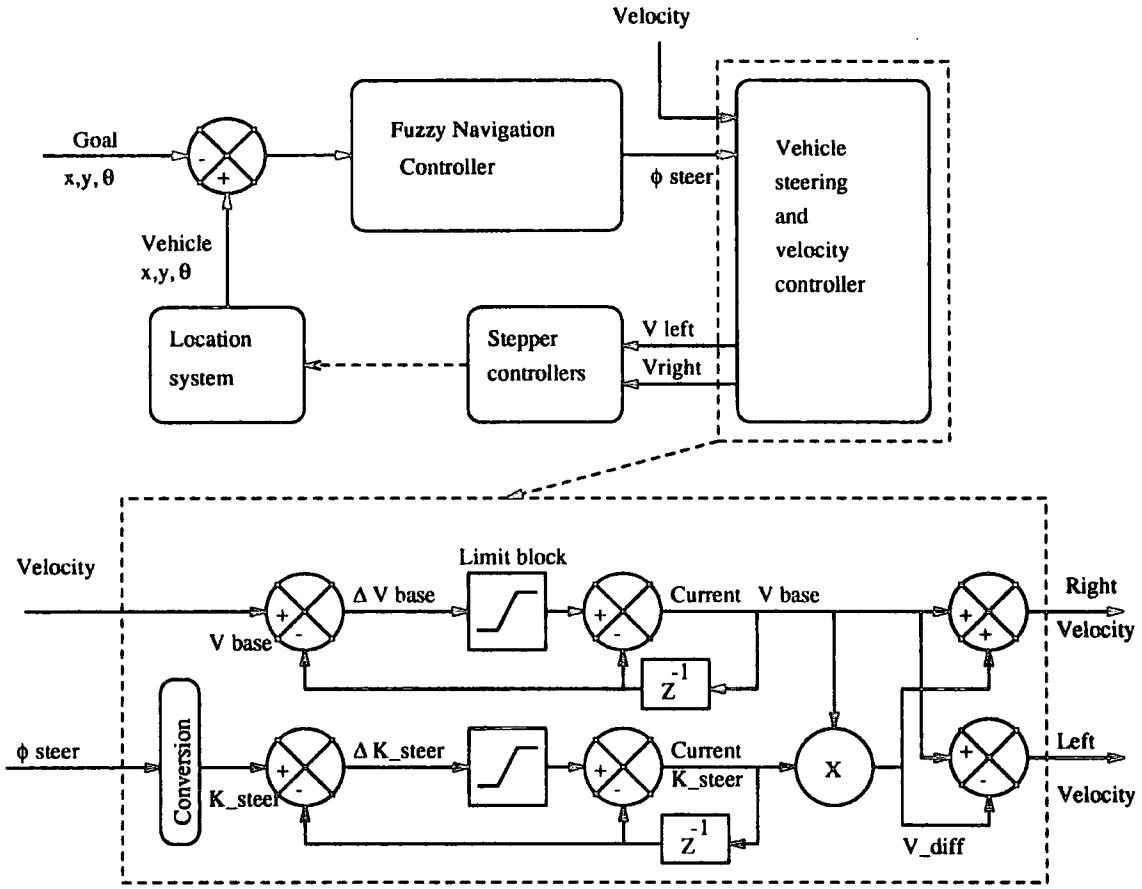

Figure 4.22: Vehicle Control Loop, Illustrating Rate Limiting Block.

4.4 Velocity Effects on Navigation

In the initial design stage the velocity of the vehicle was ignored. It was assumed that the vehicle would proceed with a uniform velocity and would be capable of achieving any given steering angle instantly without problems. Clearly any vehicle will have limits placed upon its maximum velocity and acceleration and will take a finite time to change the steering wheel(s) from one angle to another. With the steering configuration of the research vehicle large instantaneous changes in the steering angle are possible, particularly at low speeds but in order to model the use of a more conventional steering configuration, and to prevent the stepper motors stalling at high speeds, limits were applied. Figure 4.22 shows the basic control loop. The block marked 'Vehicle steering and velocity controller' was initially a simple conversion from the fixed demand velocity and demand steering angle to the required stepper velocities. As shown in the figure and explained in Section 2.5 this was expanded so as to be represented by an inner control loop. This controller imposes limits on the maximum rates of change of both the base velocity of the vehicle and the steering constant

K_{steer}.

The base velocity change limits are tightened at high steering angles while the steering change limit allows faster changes of steering angle at lower base velocities which means that certain manoeuvres, particularly those involving drastic changes in steering angle, will be affected by the base velocity. These effects are illustrated in Figures 4.23 to 4.26 which show the effects on the first two tests when the limit loop was included at vehicle velocities of 0.1m/s and 0.2 m/s. These tests were carried out using the Final controller with windowing and rule spreading operating.

In the first test it can be seen that the limiting has smoothed out the path and that in particular in Figure 4.24 the reduction of the orientation error towards zero is smoother. The effect of too high a speed is clear in Figure 4.25 showing the wide overshoot in the second test at 0.2 m/s and in Figure 4.26 where the sharp change in orientation error is replaced by a larger and larger overshoot as velocity increases. This is because the limit is effectively on $d^2\theta/dt^2$, where theta is the vehicle's orientation.

These results show that there is a need to control the velocity in order to be able to achieve rapid changes in steering angle when necessary, particularly during final docking and tight turns. It might also be desirable to be able to control velocity depending on the presence of obstacles so the vehicle would proceed at a slower speed in a cluttered area and be able to take more aggressive avoiding action. It is therefore necessary to have a fuzzy ruleset controlling not only the vehicle's steering angle but also its velocity. This modifies the control loop to the form shown in Figure 4.27.

4.5 Velocity Controller

The velocity controller is designed to increase the speed of operation of the vehicle by allowing it to perform at higher speeds when possible, that is when travelling straight and on slight curves, and to slow the vehicle down when it is turning tightly or requires a large change in the steering angle. As has been discussed in Section 4.4 the vehicle requires slowing down when a large change in steering angle is required because the maximum rate of change of steering angle is reduced at high velocities to prevent excessive angular acceleration and possible stalling of the stepper motors driving the vehicle. The controller also slows the

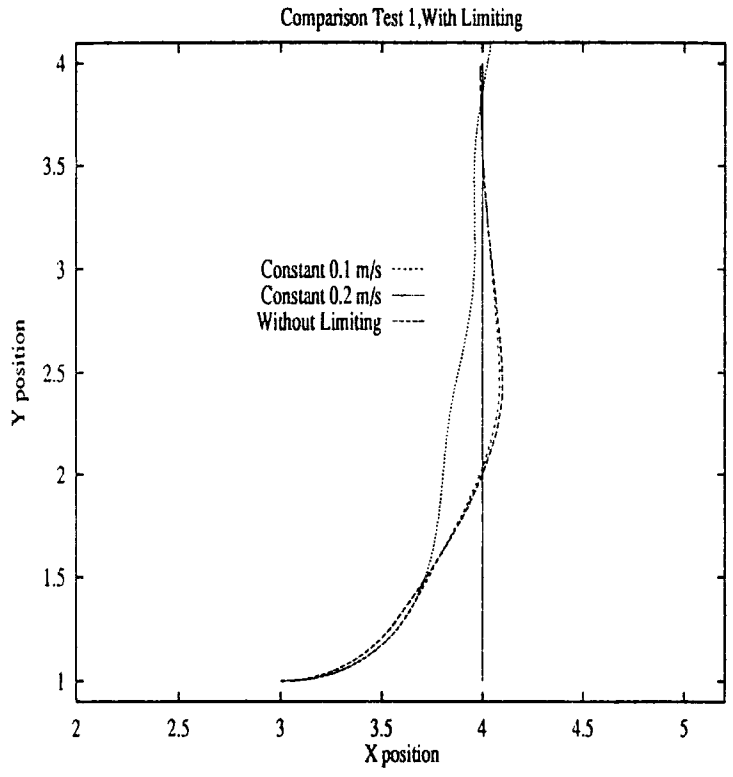

Figure 4.23: Path for Test 1, Showing Velocity Effects.

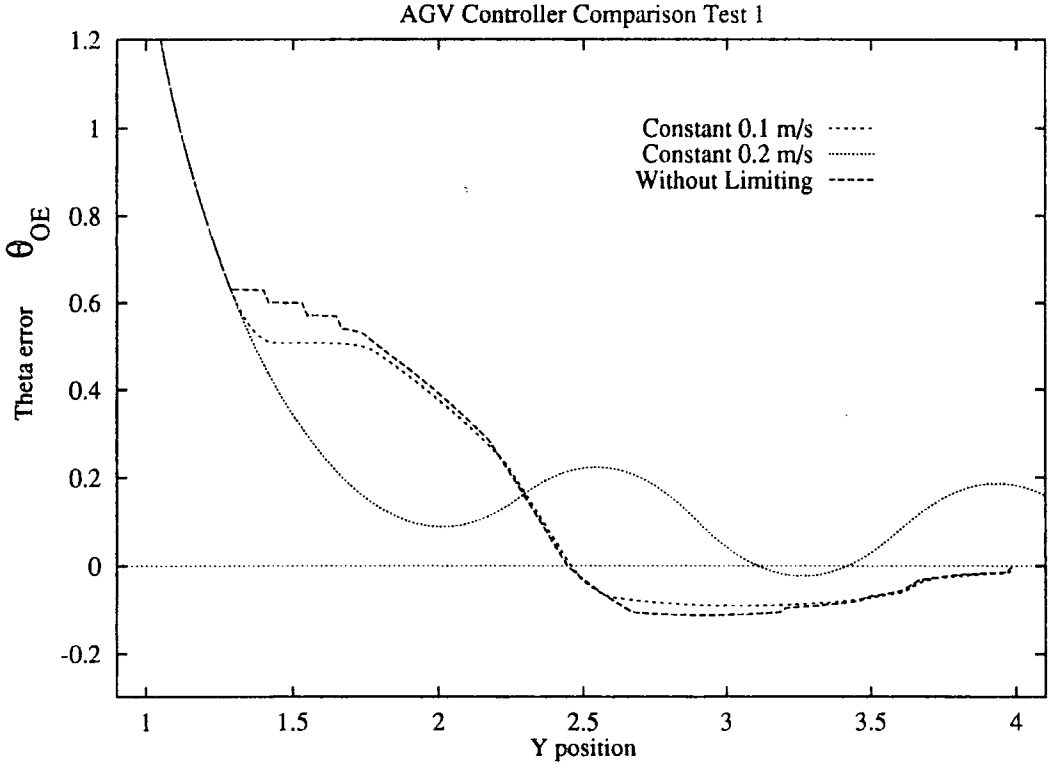

Figure 4.24: Angle Error During Test 1, Showing Velocity Effects.

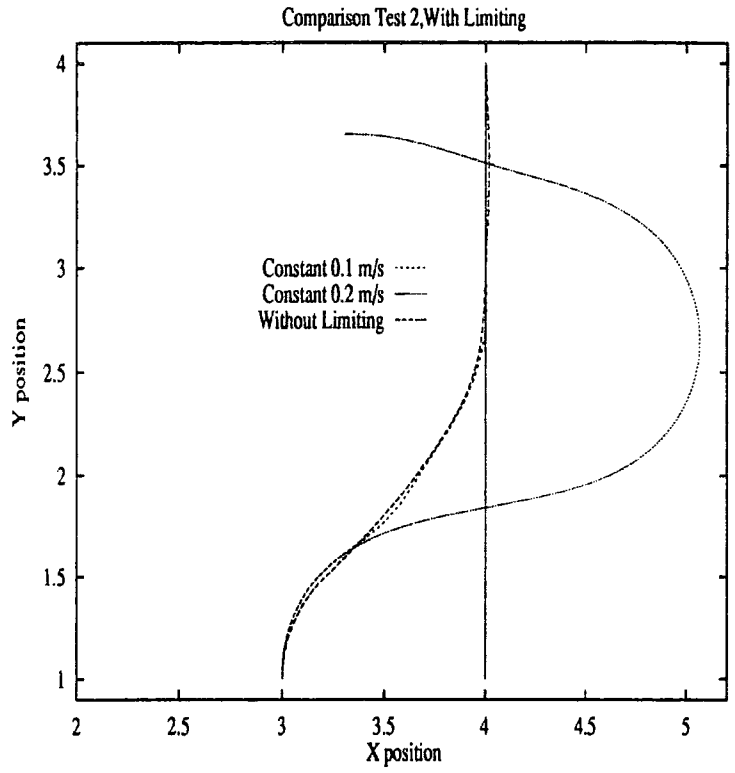

Figure 4.25: Path for Test 2, Showing Velocity Effects.

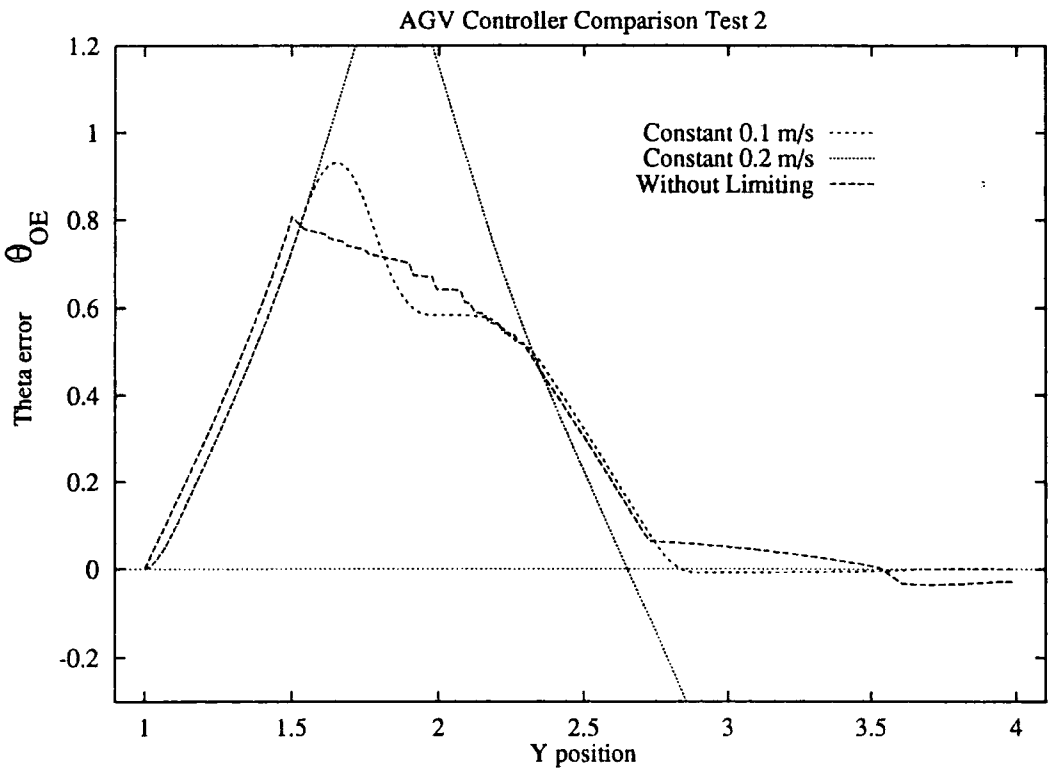

Figure 4.26: Angle Error During Test 2, Showing Velocity Effects.

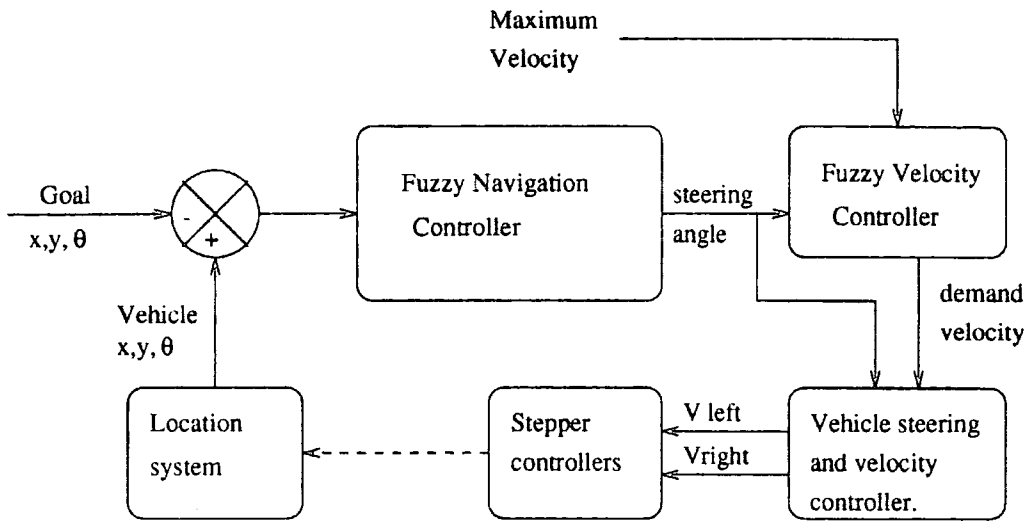

Figure 4.27: Vehicle Controller Loop, Incorporating Velocity Control.

vehicle down as it approaches the goal position.

The velocity controller is a very simple fuzzy logic controller which uses just four rules and has six input sets. The controller uses min-max inference and the correlation minimum technique to give a wider range of output values. This is achieved by using non-symmetrical output sets whose centroids vary under the correlation minimum technique depending on their degree of activation. The input and output sets are defined for three input variables, the magnitude of the demanded steering angle $|\phi|$ the magnitude of the difference between this and the actual steering angle $|\Delta\phi|$ and the distance to the goal. The sets defined for these input variables and for the output demand velocity are shown in Figure 4.28. The four rules which are used are as follows:-

IF $|\phi|$ is Large Velocity is Slow.

IF Distance is Small Velocity is Slow.

IF $|\Delta\phi|$ is Large Velocity is Slow.

IF $|\Delta\phi|$ is Small and Distance is NOT Small Velocity is Fast.

As an example of this controller in action Figures 4.29 and 4.30 show the demand and actual velocity and steering angle respectively from a vehicle test run while performing the test shown in Figure 4.33. The steps in the actual steering angle and velocity waveforms and apparent large delay are caused by the measurement method which updates slowly to ensure the measurements are taken after the vehicle has travelled a large enough distance to give an accurate result. These graphs show the smoothing effect which the limit block has

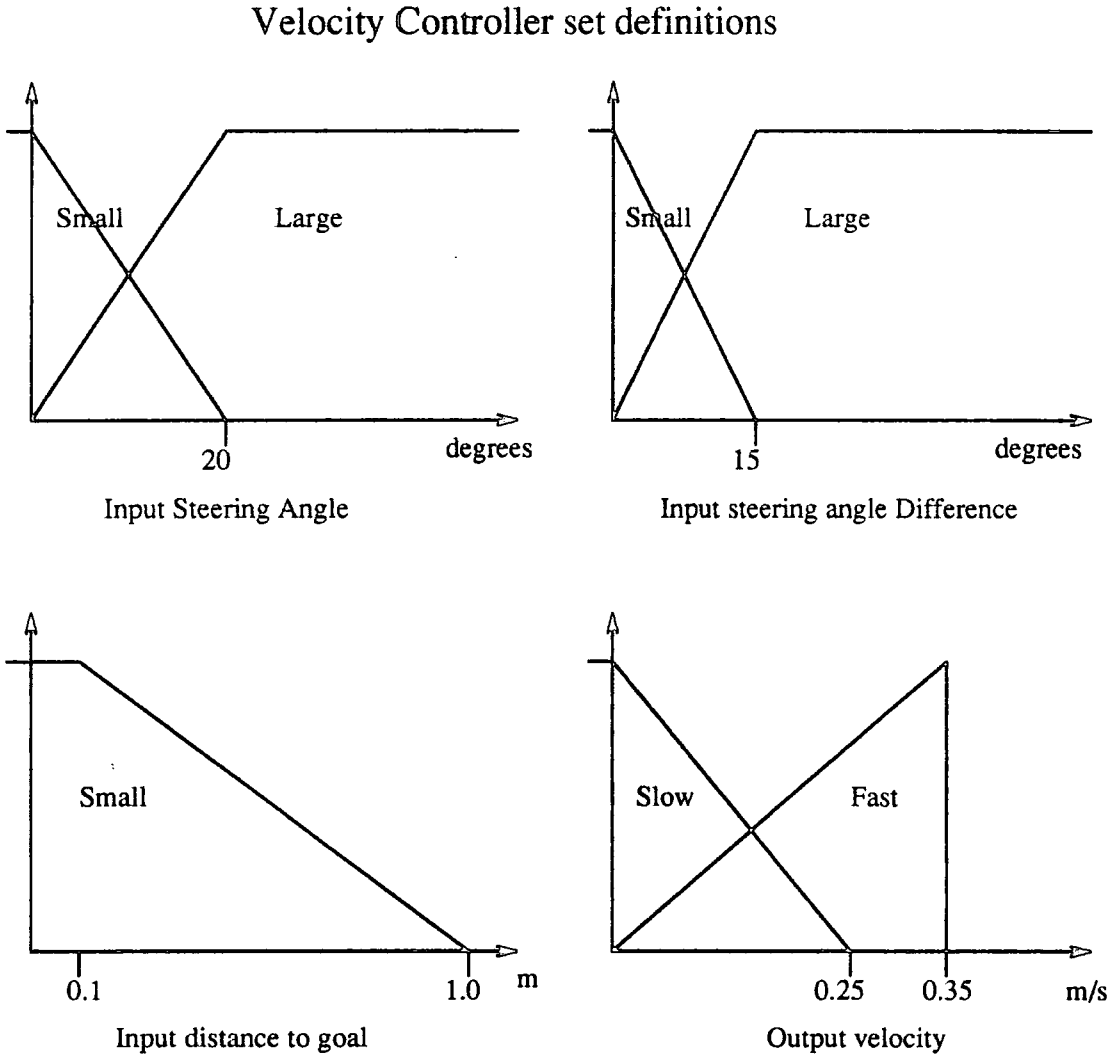

Figure 4.28: The Set Definitions used in the Velocity Controller

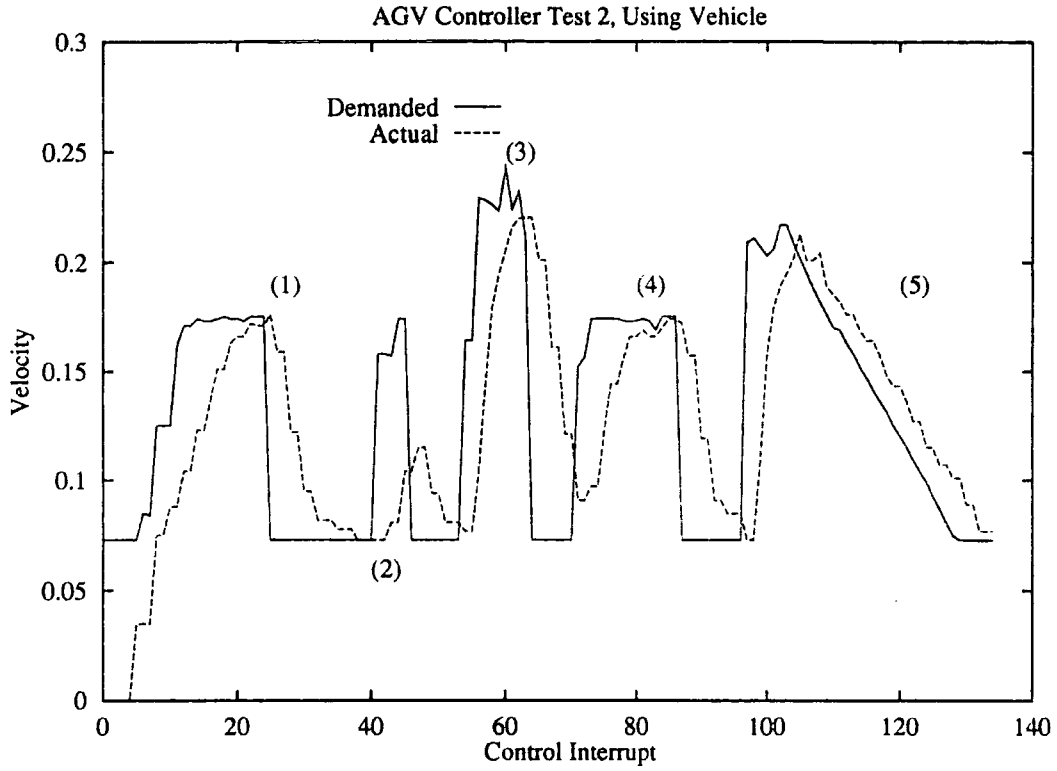

Figure 4.29: Demanded and Actual Velocity of AGV During Test 2

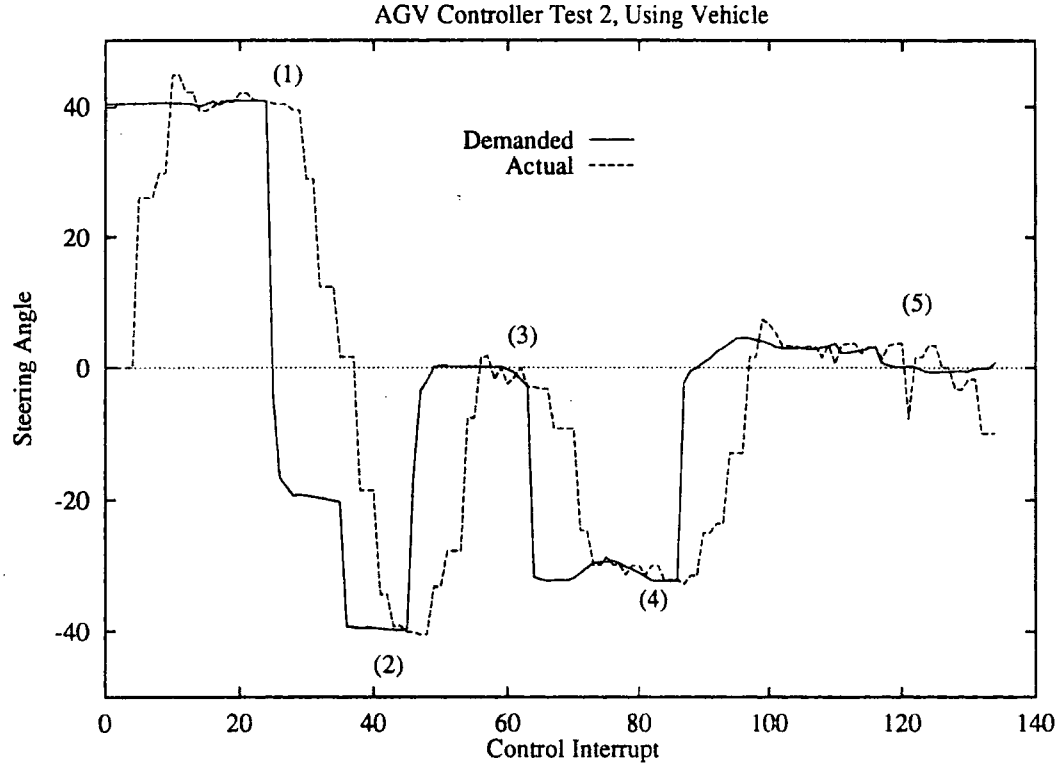

Figure 4.30: Demanded and Actual AGV Steering Angle During Test 2

on the changes in steering angle and velocity as well as illustrating how the vehicle slows down when a large change in steering angle is seen and as the vehicle approaches the goal. The vehicle accelerates to approximately 0.175 m/s at the point marked 1 where it can be seen it has achieved the desired steering angle of plus 40 degrees. It then rapidly decelerates due to the large change in steering angle, to the opposite extreme of minus 40 degrees which is achieved at point 2. The vehicle then straightens up and the demand velocity increases to a maximum of 0.25 m/s when the vehicle is continuing in a straight line around the point marked 3 on the graph. It can be seen by comparing points 3 and 4 that the overlap in the controller which activates the Slow velocity set for a Large steering angle as well as Fast velocity once that steering angle has been achieved gives a higher velocity when the vehicle is travelling in a straight line than when it is on a tight curve. The reduction in speed as the vehicle approached the goal is shown by the section of curve marked 5 when the vehicle is travelling straight ahead but it slows down in preparation for stopping at the goal position.

Figures 4.31 to 4.34 show that the path of the vehicle over the first two comparison tests is much the same with the velocity controller active as it was at a constant $0.1m/s$ but the time taken is shorter since the vehicle is able to accelerate to higher speeds when it is travelling in a straight line.

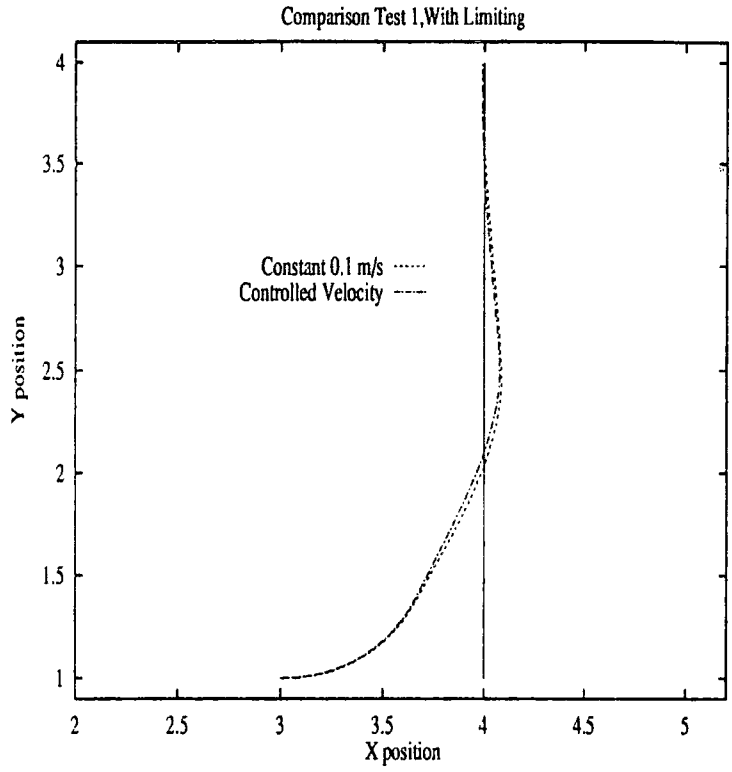

Figure 4.31: Path for Test 1, Using Velocity Control.

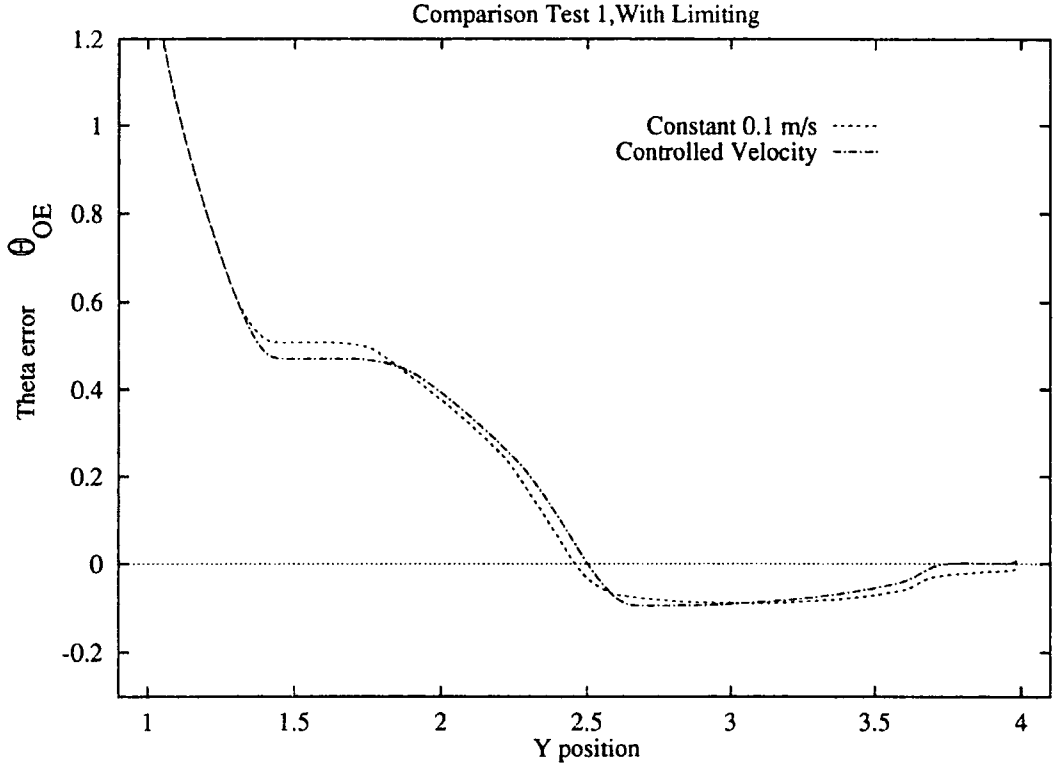

Figure 4.32: Angle Error During Test 1, Using Velocity Control.

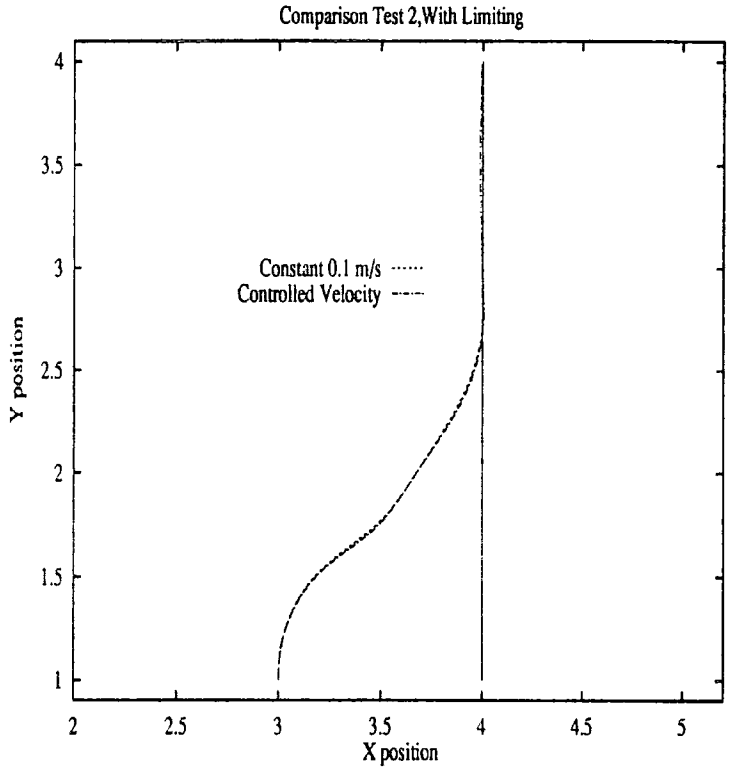

Figure 4.33: Path for Test 2, Using Velocity Control.

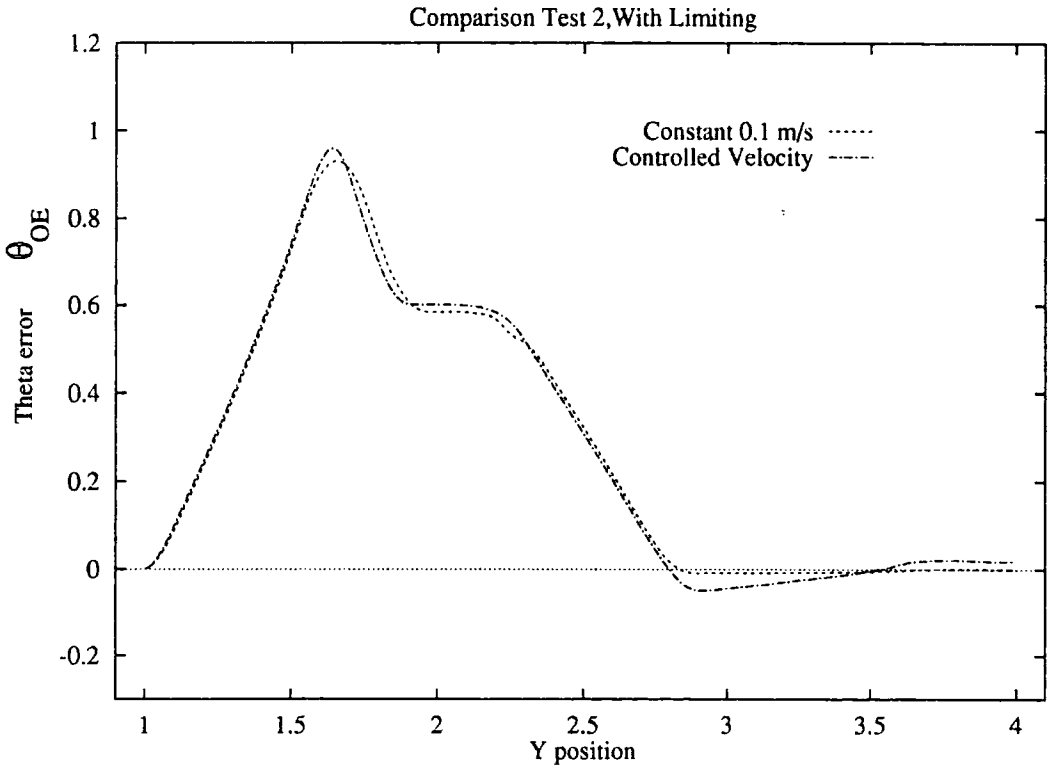

Figure 4.34: Angle Error During Test 2, Using Velocity Control.

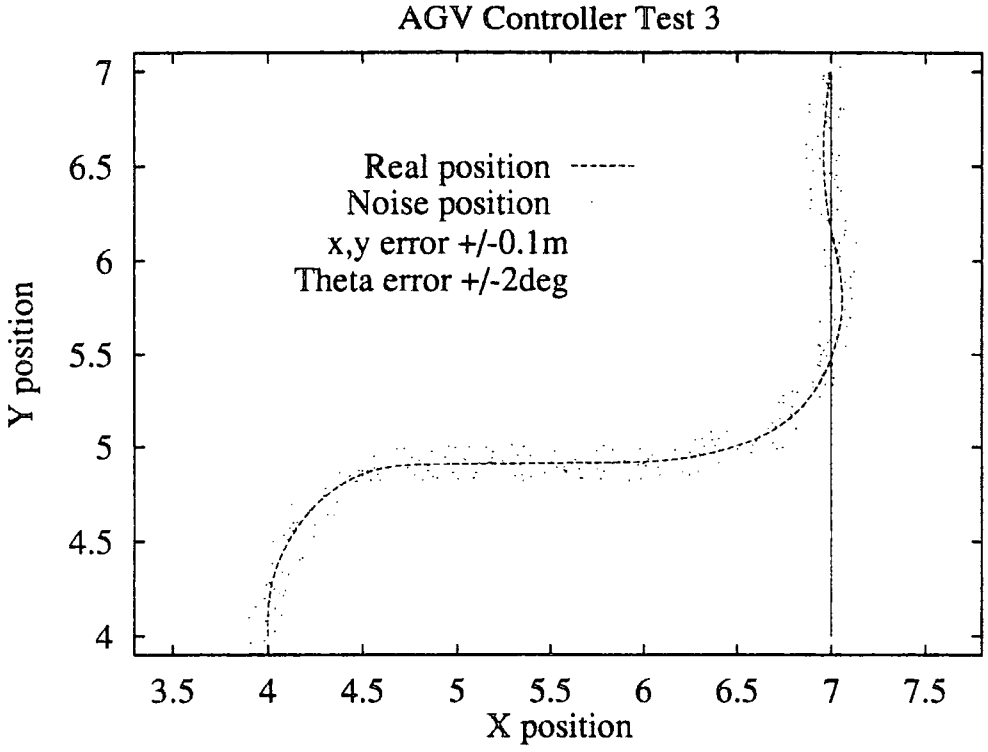

Figure 4.35: Path for Test 3, Low Level Noise.

4.6 Noise Effects on Navigation

It is often claimed that fuzzy logic control gives good performance even in the presence of noise. To examine how the controller coped with uncertainty in the vehicle's position a noise generator was added to the simulation. At each control interrupt a random displacement was added to both the X and Y positions of the vehicle and to its orientation. These random factors were uniformly distributed over a set range with zero mean. Figure 4.35 shows the path of the vehicle with up to plus or minus ten centimeters of displacement error and plus or minus two degrees of orientation error performing test three. The dots show the position passed to the controller while the line shows the true path of the vehicle. It can be seen that the vehicle still manages to achieve its goal with this level of noise. With more noise as shown in Figure 4.36 the path is much more degraded but still achieves the goal. The controller is therefore seen to be robust to random errors in positioning and orientation, although it will still be vulnerable to systematic errors produced by faulty positioning over many control interrupts.

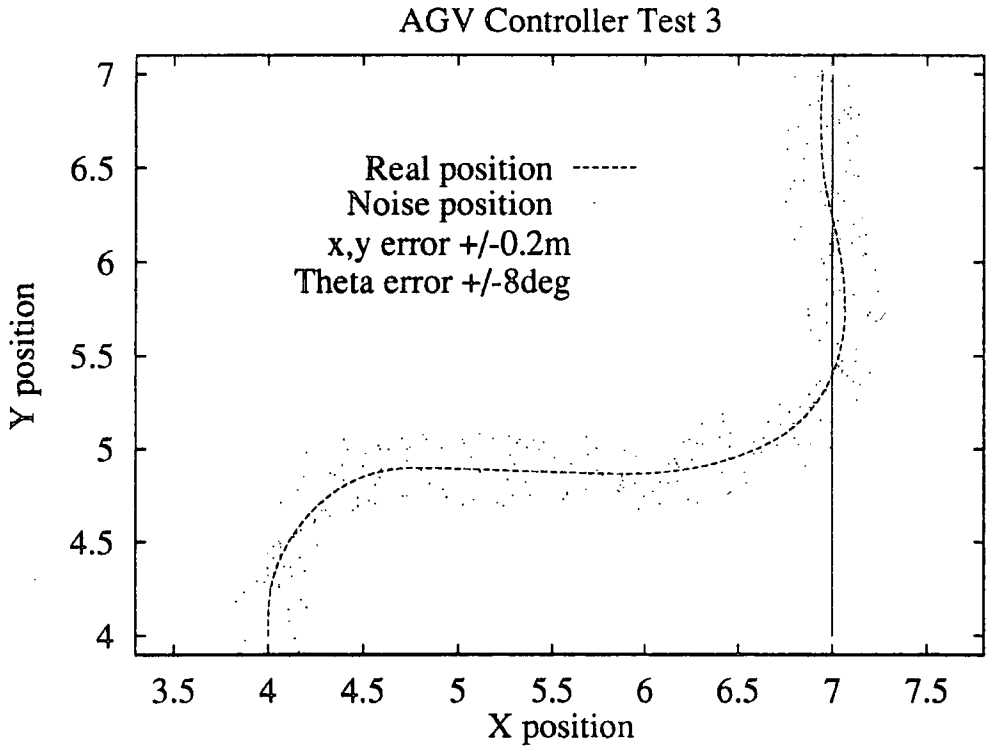

Figure 4.36: Path for Test 3, High Level Noise.

Chapter 5

Obstacle Avoidance Controller

This chapter is concerned with the development of the obstacle avoidance function for the vehicle and how this is combined with the fuzzy logic based navigation controller. The principles of obstacle avoidance are discussed and the particular considerations which lead to the final design are introduced. The representation of sensor data is briefly discussed along with possible ways of implementing obstacle avoidance using fuzzy logic rules. The basic method is introduced and the algorithms used to design avoidance sets and to calculate the activation of those sets from knowledge of the environment are explained.

The combination of the obstacle avoidance rules and navigation rules required the development of some new techniques, rule spreading to infer alternative rules from existing ones and sliding window defuzzification to prevent collisions caused by the indecision introduced by other defuzzification methods. These new techniques are fully described and their effects examined. Finally a detailed analysis is given of a test run in which the vehicle has to avoid obstacles in its path before being able to reach its goal.

5.1 Obstacle Avoidance Principles

A problem which all autonomous vehicles must overcome is that of unwanted obstacles in their path. The first priority is simply to prevent the vehicle from colliding with an obstacle. The action which is taken to avoid an obstacle should be such that it does not conflict with the overall navigational goal and in an ideal system the navigation controller would be able

to formulate a new control strategy based on the positions of known obstacles.

When a physical vehicle is used rather than a simulation it has been found that separately designed and implemented techniques for navigation and obstacle avoidance can produce problems as the two functions are not in reality independent. Payton et al [37] propose a 'fine grain' approach to the design of AGV control systems in which different subsystems (such as navigation and obstacle avoidance) produce several suggestions as to the desired control output. The final output would then be based on a comparison of these suggestions to produce the most suitable result for differing goals. This type of procedure is closely analogous to a fuzzy logic system where inputs activate the output rules to varying degrees and the final result is based on a combination of all these rules.

It is also important to consider the physical size of a vehicle and the constraints on its motion. In many examples of avoidance systems the vehicle is assumed to be cylindrical and obstacle sizes are expanded to allow the problem to be considered as the motion of a single point. In common with most factory load carrying vehicles the research vehicle is rectangular and it is important to note whether the vehicle is front or rear wheel steered because the relative motions of the sides of the vehicle will be different. In developing an avoidance scheme it is important therefore that the method be linked to the geometry of the vehicle and to the limits of its steering performance. An ideal method would not affect the normal motion of the vehicle unnecessarily but would guarantee that the vehicle avoids collisions.

5.2 Obstacle Representation

Obstacle avoidance systems are heavily dependent upon the sensors used to detect objects around the vehicle. Sensors can often miss objects or produce false indications of an object where none exists in reality. To represent this sensors need to be characterised so that the information they give can be assessed for accuracy. An established technique for representing sensor derived information about obstacles is the *certainty grid* method used by Elfes [18]. This method divides the area around the vehicle up into a grid and assigns an occupancy value to each square on the grid to indicate the probability that there is an object in that grid square.

This method has several valuable properties. Firstly it can fuse the data from several sensors to give a more accurate picture of the surrounding area. Secondly the information held in the grid can be used by routines for navigation and obstacle avoidance without needing to know how the information was acquired, thus improving portability. Thirdly the information being of a probabilistic nature would seem to be ideal for processing by a form of probabilistic or fuzzy logic.

5.3 The Form of Avoidance Rules

There are two different techniques which could be used to add an obstacle avoidance method to the fuzzy navigation controller. One would be to have rules of the form :-

IF obstacle AHEAD LEFT then steer POSITIVE MEDIUM

which would give activation to a set designed to steer the vehicle away from the obstacle. There are some problems with this method which could cause the vehicle to hit an object even though it had been detected or deteriorate the performance of the navigation controller.

The activation of a set to steer in a specific direction would not guarantee that the vehicle would take that path since if this output set were combined with the activated output sets from the navigation controller the defuzzified output could give a result that turned towards rather than away from an obstacle. To avoid this problem a very large weight could be applied to the obstacle avoidance rules so they dominated the response but the vehicle would then simply shy away from obstacles most of the time and be lacking in guidance to the goal. A large weighting would also compound another problem which is the effect of an object which the suggested output from the navigation controller would avoid. Using the above example if the navigation control rules already had an output of POSITIVE LARGE the combination with the obstacle avoidance rule would in fact bring the vehicle closer to the obstacle.

The alternative to adding new rules is to affect the output already generated by the navigation control rules. In this case rules take the form :-

IF obstacle AHEAD LEFT then DON'T steer NEGATIVE SMALL

This is an *inhibitive* rule which reduces the activation of an output set rather than increasing it. This has the advantage that if the navigation controller would have missed the obstacle anyway the rule will not alter this and if the inhibited set was active the other activated sets can contribute to a more sensible output which will help to achieve the goal.

The aim of these inhibitive rules therefore is to produce a *mask vector* which can be multiplied with the fuzzy fit vector to give an overall output vector. The mask vector contains values indicating how acceptable each possible steering angle set is in view of the proximity of obstacles. An entry of zero in the mask vector indicates a nearby obstacle in that direction and will prevent any activation of the steering set which would take the vehicle towards the obstacle. A value of one in the mask vector indicates that there are no obstacles in that particular direction. Values between 0 and 1 indicate either a possibility of an obstacle or that a confirmed obstacle is a long distance away in that direction.

5.4 Definition of Avoidance Sets

The input sets for the obstacle avoidance controller are linked to particular areas of the certainty grid relative to the vehicle. The area covered by a set is the area which the vehicle would occupy if a particular output set was activated. Each rule links one or more of these input sets to a degree of inhibition of the corresponding output steering angle set which ranges from 0, total inhibition, to 1, no inhibition. Sets covering areas close to the vehicle have the greatest inhibitive effect and prevent any activation of their linked output set while more distant sets merely reduce the activation. The activation of an input set is defined as the maximum occupancy value found within the area defining the set which is effectively a fuzzy OR of all the occupancy values within the area. This activation value combined with the inhibition value of the rule is used to calculate the entry in the output mask vector corresponding to the output set which the rule acts upon. The completed mask vector is then ready to be combined with the output of the navigation rules to give the final output steering angle.

The areas covered by the avoidance sets have been the subject of much experimentation as has the inhibition value of the sets and are covered in Section 6.6. The minimum width of an avoidance set is relatively easy to define as it must be wide enough to ensure that the whole vehicle will remain inside the set area as the vehicle moves on the curve specified

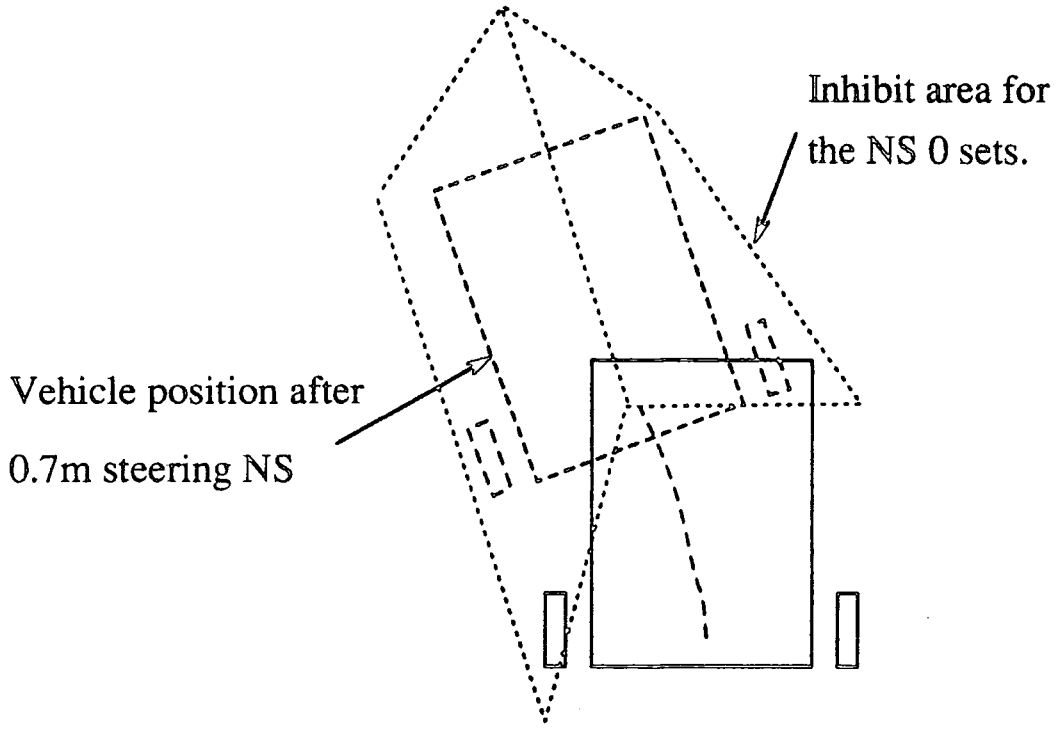

Figure 5.1: Avoidance Set Area for 'Negative Small' Steering Angle.

by the output set. It is harder to define a length for the set, that is the distance over which the vehicle should be able to travel and remain within the set. Clearly for the zero inhibit value a minimum set size should be such that the vehicle could stop within the set so the minimum length would be the distance travelled within one control interrupt plus the vehicle's stopping distance. A further consideration is that it is important to enable the vehicle to look far enough ahead to be able to avoid situations where although there is no problem over a short distance the vehicle would be unable to avoid an obstacle further on if it continued on that path. For instance entering a narrow corridor which is blocked at the far end. Although the vehicle may not immediately have a collision if it enters the corridor it will be unable to steer left or right.

As a starting point very pessimistic sets were hand generated by considering the vehicle's geometry and steering angle sets. The total inhibition sets were designed so as to cover the area which the vehicle would occupy after travelling one vehicle length plus the clear space needed in front of the vehicle after this distance had been travelled to ensure further movement would be possible.

To illustrate this the avoidance area sets for NS, a slight left hand turn, are shown in Figure 5.1. The solid outlined vehicle shows the reference position of the vehicle while the

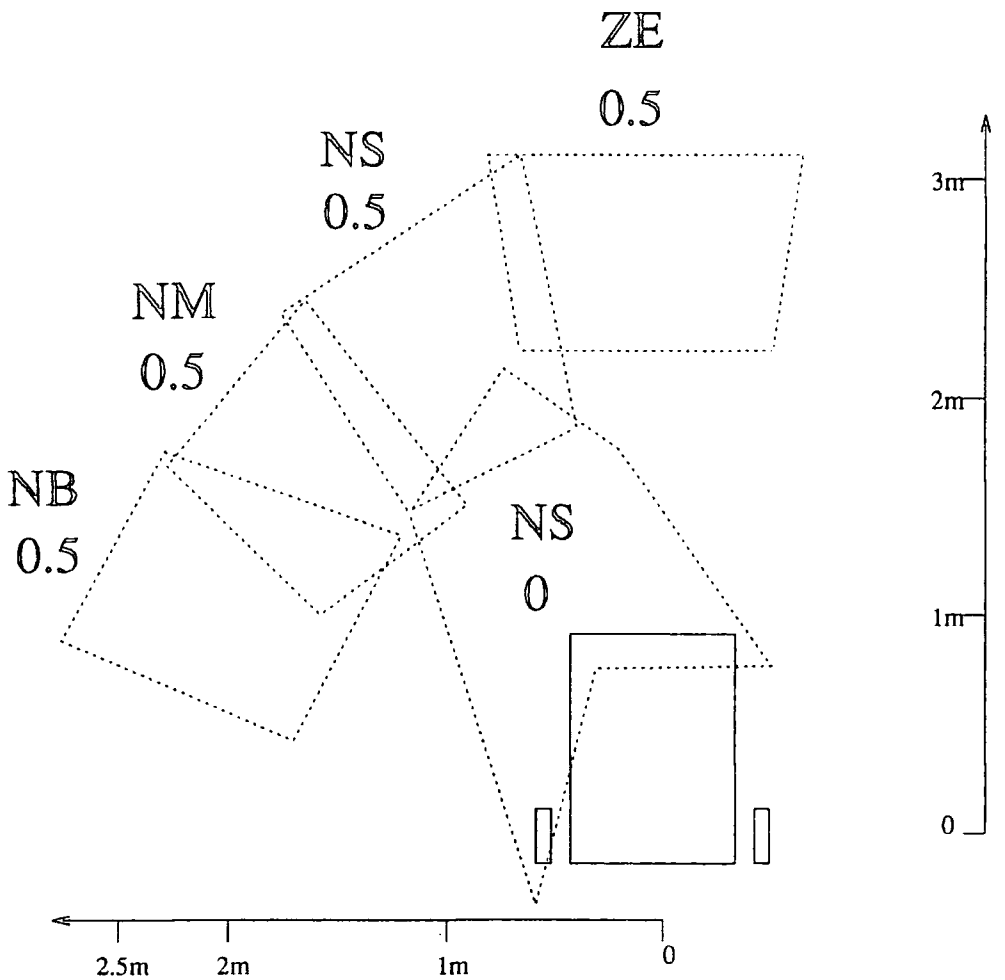

Figure 5.2: Distant Avoidance Set Areas for all Steering Angle Sets.

dashed outlined vehicle is the position the vehicle would be in if it continued for 0.7m with a negative small (18° left) steering angle. The dotted outline is the area of the certainty grid which would be searched for obstacles and which would induce a zero entry in the mask vector for the negative small steering angle. This area includes the vehicle position and any positions it would have passed through to get there. This means that if the area is found to be clear the vehicle can continue for at least 0.7m with a negative small steering angle without risk of collision. In order for the sets to be easy to map to the occupancy grid and searched for objects each set is stored as the four corners of a convex quadrilateral. The area covered by the NS total inhibit set is therefore stored as two of these shapes. There may therefore be several quadrilateral shaped areas which must be searched to discover the lowest inhibition value which will be the entry in the mask vector for a particular output set.

Further sets were generated with the inhibition level of 0.5 to discourage steering towards

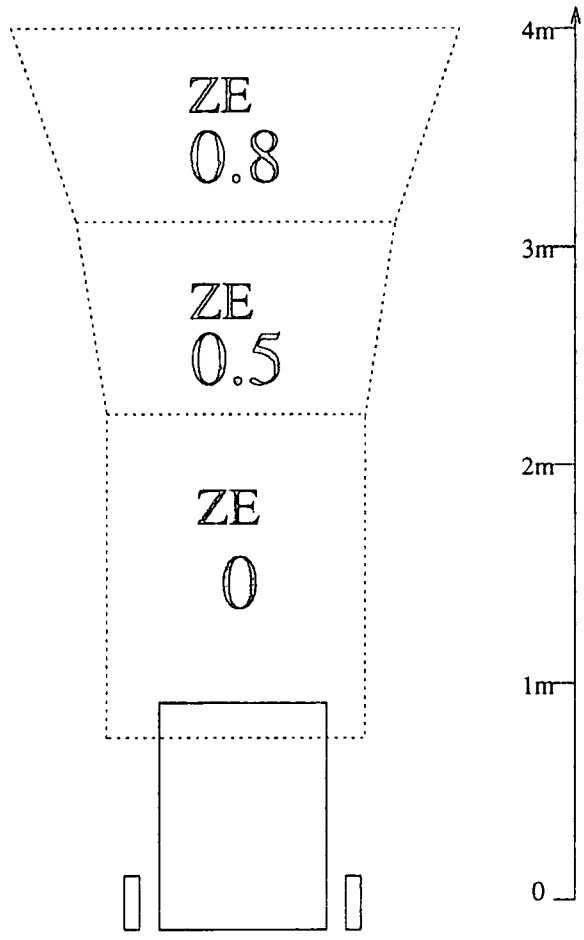

Figure 5.3: Avoidance Set Areas for 'Zero' Steering Angle.

distant objects covering the space the vehicle would occupy if it travelled straight ahead one vehicle length from the end of the total inhibiting set. Figure 5.2 shows these areas relative to the current position of the vehicle. It can be seen that the areas form a semi-circle in front of the vehicle so providing early warning of likely problem directions. In an attempt to induce early avoiding action of obstacles directly in front of the vehicle an additional set with an inhibit value of 0.8 was introduced, covering the area for half a vehicle length in front of the 0.5 inhibition set linked to the output set ZE. All the avoidance set areas linked to this steer straight ahead output set are shown in Figure 5.3.

5.5 Calculating the Activation of Avoidance Sets

The avoidance sets are defined relative to the local origin of the vehicle, that is the centre of the rear wheelbase, while the occupancy grid is defined relative to the global origin. The first

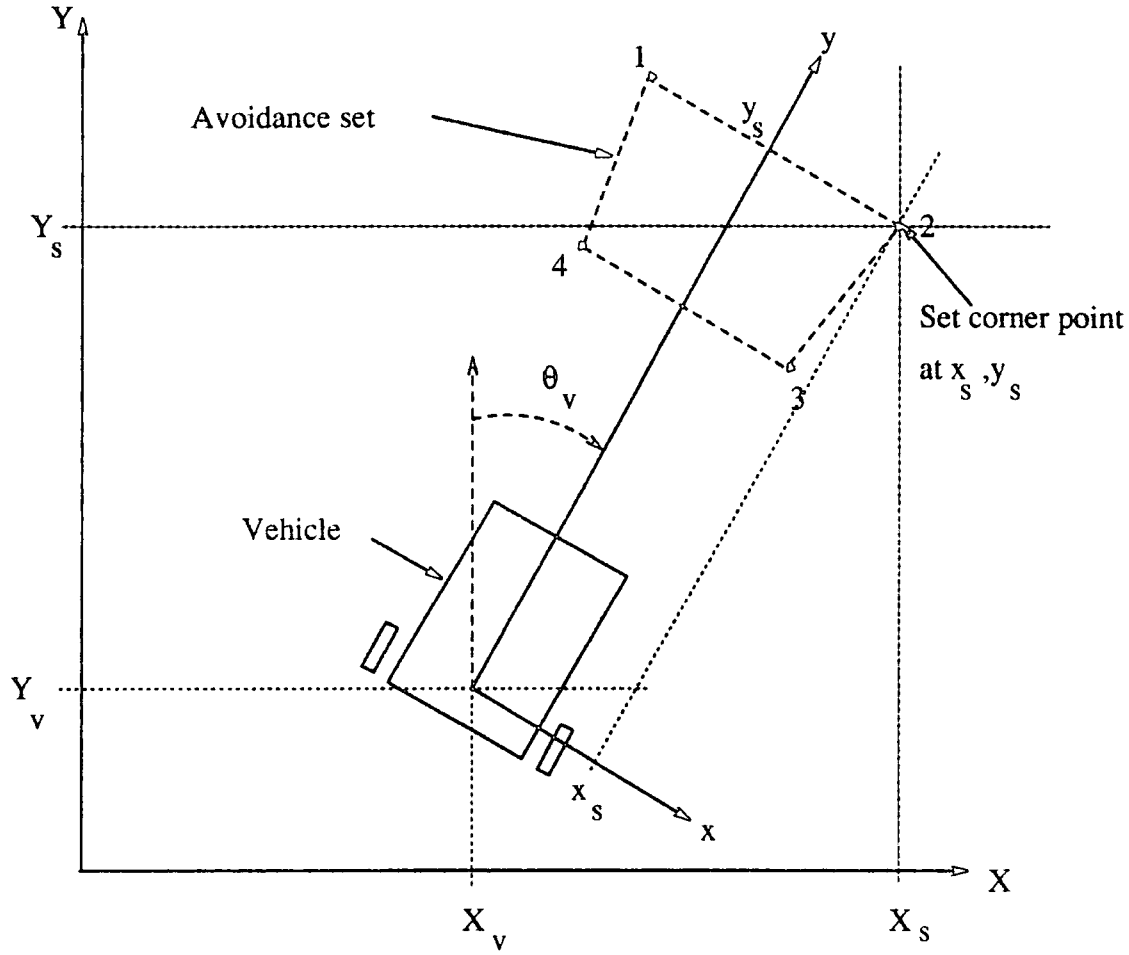

Figure 5.4: Avoidance Set Area in Global and Local Co-ordinates.

step is to map the avoidance set area onto the grid before it is searched. Figure 5.4 shows a typical situation where the vehicle is at the global position X_v, Y_v with an orientation of θ_v. The global coordinates X_s, Y_s of a corner of the set which is at x_s, y_s in the local co-ordinate scheme are found from Equations 5.1 and 5.2.

$$X_s = X_v + y_s \sin\theta_v + x_s \cos\theta_v \tag{5.1}$$

$$Y_s = Y_v + y_s \cos\theta_v - x_s \sin\theta_v \tag{5.2}$$

These coordinates are then divided by the size of the grid squares (currently 0.1m) to yield a position within the array of occupancy values. The problem now becomes that illustrated in Figure 5.5 where a shape is defined on a grid of squares and it is desired to search all squares which are totally or partially included inside the shape. Thus all the shaded squares in the figure must be searched to find the maximum occupancy value within

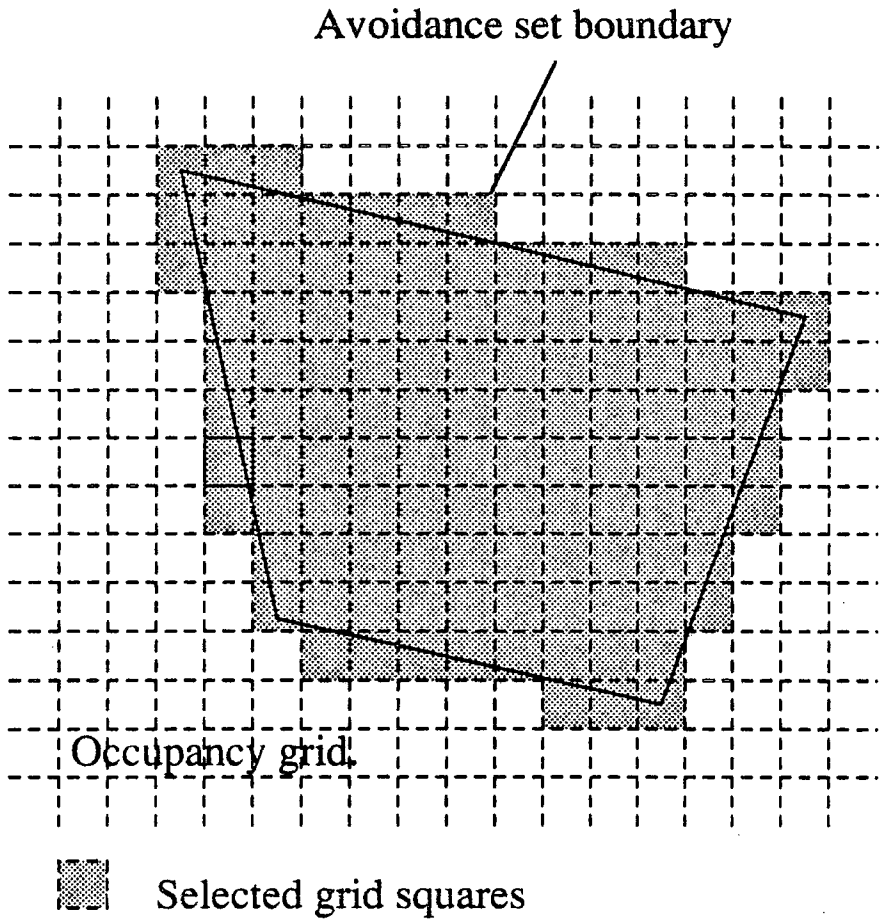

Figure 5.5: Avoidance Set Area Mapped onto an Occupancy Grid.

the avoidance set area. The occupancy grid has its origin in the bottom left corner of the grid so each square is addressed by the coordinate of its bottom left corner. To find the maximum possible extent in Y of the included squares the corners with the maximum and minimum Y ordinates are found and rounded down to give integer row values of the occupancy grid. To reduce the computation time in the search routine the four corner points are stored in clockwise order for the negative and zero steering sets and anti-clockwise order for the positive steering sets which are derived by reflection about the vehicle's centreline of the negative set area definitions. This means the routine always knows which points relative to the minimum Y point define the current boundary lines.

The way in which the routine to search the desired area of the grid works is illustrated by Figure 5.6. The avoidance set area is searched row by row, starting with the row with the minimum Y value and continuing until the row including the uppermost corner of the avoidance area set is reached. The routine finds the range in X to be searched for any given row Y_n by keeping track of the set boundary, tracking it clockwise for the minimum

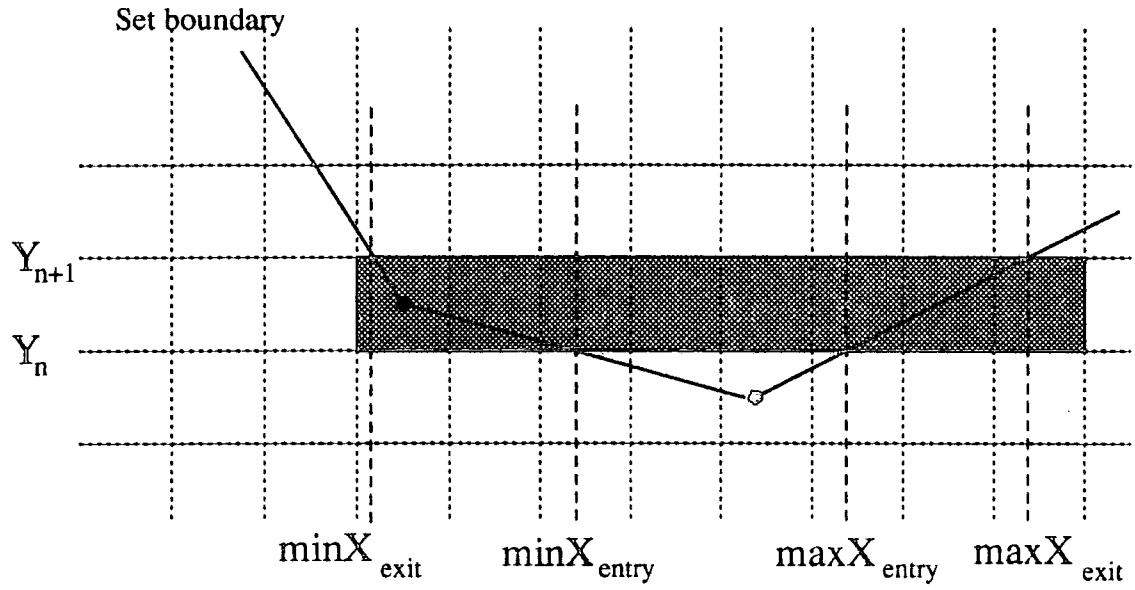

Figure 5.6: Finding the Edges of a Set on the Grid.

X values and anti-clockwise for the maximum X values.

For any given row, Y_n the routine finds the X values at which the maximum and minimum boundaries intersect Y_n and the next row Y_{n+1}. For the minimum X boundary point as shown in Figure 5.6 the intercept with Y_n is $\text{minX}_{\text{entry}}$ and with Y_{n+1} $\text{minX}_{\text{exit}}$. The X start value is derived from the minimum X value of the intersection points and any corner points between Y_n and Y_{n+1}. In the example given it is clearly $\text{minX}_{\text{exit}}$ which is chosen. Similarly the X finish value is derived from the maximum X value of the intersection points $\text{maxX}_{\text{entry}}$ and $\text{maxX}_{\text{exit}}$ and any corner points between Y_n and Y_{n+1}. This gives the X range of the grid squares to be searched which are shaded on the diagram. The occupancy values of all included squares on the row are then found and compared to the current maximum occupancy value of the set. [1] Once the whole set has been scanned the overall acceptability is calculated from Equation 5.3. This is compared to the current entry in the mask vector for the associated output set and if it is a smaller value it is placed in the mask vector. The graph of Equation 5.3 can be seen in Figure 5.7 and it can be seen that it ensures that the lowest value which can be entered in the mask vector is the overall inhibition value of the set and occurs when there is an 100% occupancy chance. The acceptability value entered in the mask vector rises linearly with decreasing occupancy percentages until it reaches 1 for entirely unoccupied areas. This equation was adopted for the initial testing because of

[1] To conserve memory occupancy values are stored as integers between 0 and 100

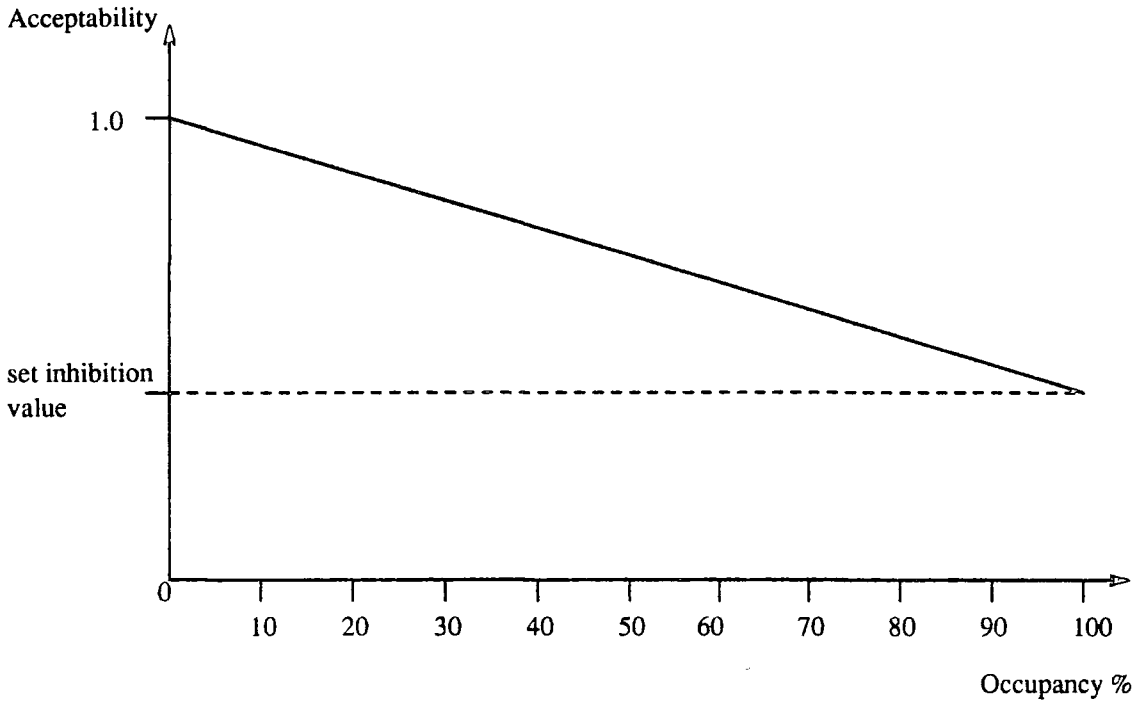

Figure 5.7: Graph Illustrating Calculation of Acceptability.

its simplicity but there are many possible alternatives as discussed in Section 8.3

$$\text{Mask Value} = 1 - (1 - \text{Set inhibit value}) \times (\text{Maximum Occupancy}/100) \qquad (5.3)$$

Any set which is found to have part of its area outside the bounds of the grid is assigned the maximum occupancy value. To speed searching of the sets the routine ignores any sets whose inhibition value is higher than the current mask value and terminates a search as soon as a 100% occupancy value is found. The routine searches all the sets to fill the mask vector which modifies the output of the navigation controller.

5.6 Conflicting Outputs Can Produce Collisions

There are some problems with using inhibitive rules. As noted in Section 4.2 conflicting rules can give a defuzzified output which is a member of none of the activated sets so even if a set were prevented from being active a member of that set could still be selected. Also if all the activated sets were reduced to zero activation by obstacle avoidance rules there

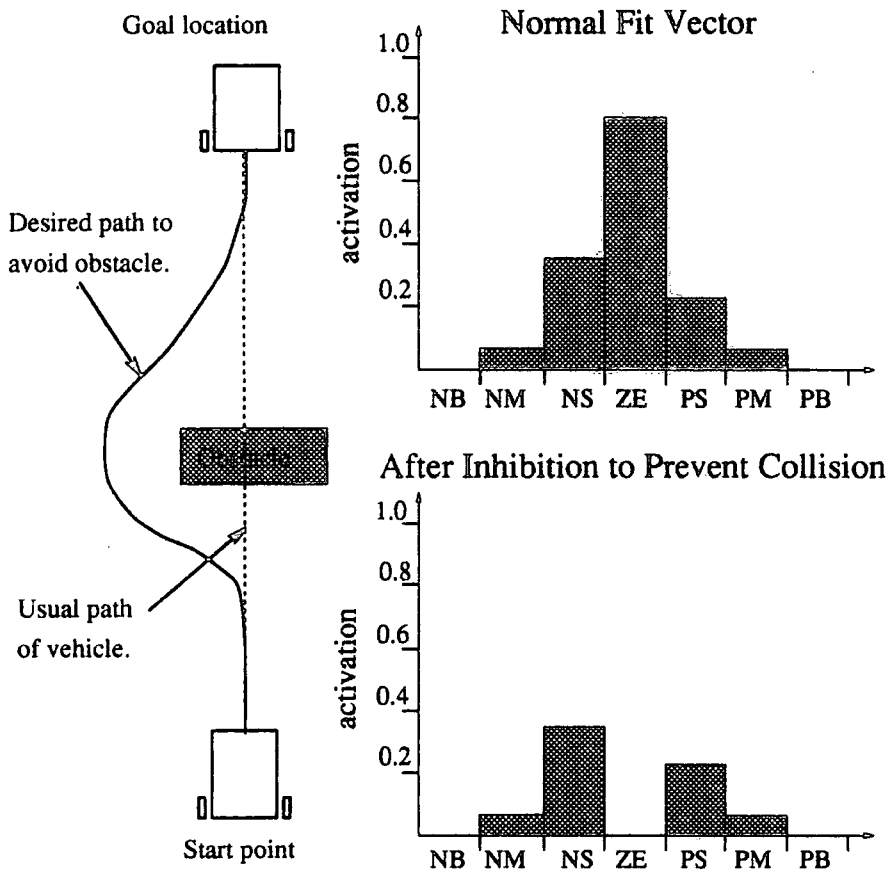

Figure 5.8: Fuzzy Fit Vector Before and After Inhibition to Avoid a Possible Collision.

would be no controller output. Section 5.7 deals with the techniques developed to overcome these limitations.

Considering the situation shown in Figure 5.8 the controller would normally drive the vehicle straight ahead to reach its goal but its path is blocked by an obstacle. The obstacle avoidance system inhibits the ZE, steer straight ahead, output. However with centre of gravity defuzzification the resultant output would still be to steer straight on as a compromise between going left or right round the obstacle!

5.7 New Fuzzy Control Techniques

To guarantee that the defuzzified output will not fall within one of the inhibited sets a new method is used which has been called *sliding window defuzzification* or *windowing*. To provide alternative suggestions if all the output sets activated by the navigation controller are inhibited by the obstacle avoidance controller a technique called *rule spreading* has been

devised. These new techniques have been described in [51] and [52].

5.7.1 Sliding Window Defuzzification

To use an obstacle avoidance routine which reduces or removes the activation of output sets it must be possible to predict the effect that the inhibition will have on the defuzzified output. As mentioned in Section 4.2 this is not possible with centre of gravity defuzzification so a different technique has been used called *sliding window defuzzification* or *windowing*. This technique is a cross between maximum-membership defuzzification and centre of gravity defuzzification. The technique involves locating the set with the maximum activation and then applying centre of gravity defuzzification to this set and its immediate neighbours. In effect windowing modifies the fuzzy fit vector in the manner shown in Figure 5.9 before centre of gravity defuzzification is performed. The activation of sets not greater than the window width from the most activated set are included while the others are ignored. This can be expressed in Equation 5.4 which maps the normal fuzzy fit vector F, in which set m has the maximum activation, to the windowed fuzzy fit vector W by applying a window of width w.

For the i'th entry in the fuzzy fit vector

$$W_i = \begin{cases} F_i & \text{if } |i - m| \leq w \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

This results in a fuzzy fit vector with a maximum of $2w + 1$ non-zero entries. With a window width of zero, which includes only the most activated set, the method is effectively maximum-membership defuzzification while if the window width is increased so it includes all the sets it is the same as the standard centre of gravity defuzzification.

In order to be able to guarantee that the defuzzification process will not select an output value which has been banned by the obstacle avoidance routine it is necessary to use only the set with the maximum activation and its two neighbours on either side. This reduces the number of sets which influence the output but as tests have shown has little practical difference since the activated output sets are usually grouped around the most activated set.

The effect which windowing has in the example shown in Figure 5.8 can be seen in

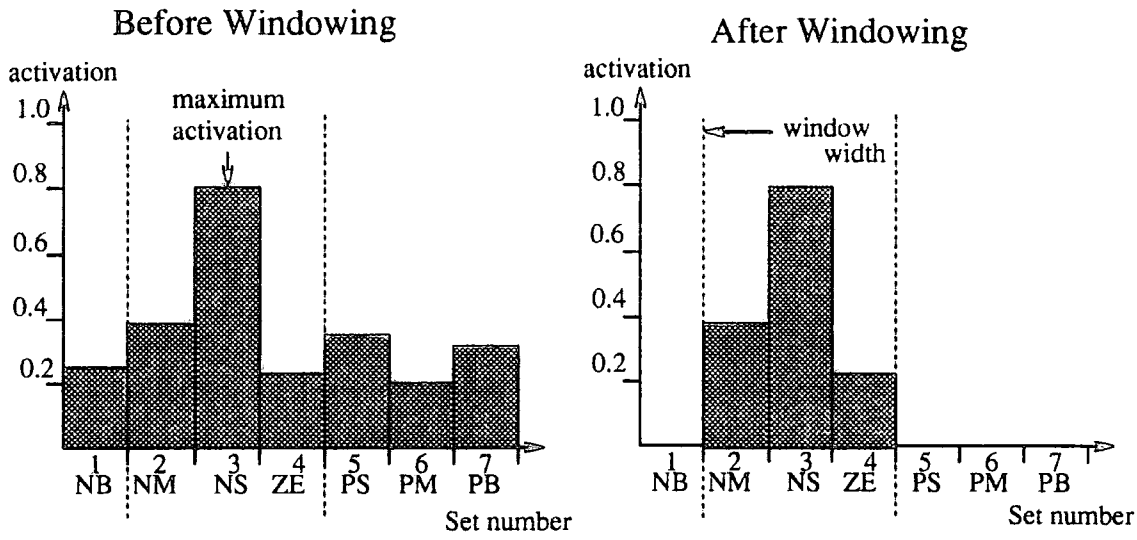

Figure 5.9: The Effect of Windowing On the Fuzzy Fit Vector.

Figure 5.10. It can be seen that the ZE steering rule has been totally inhibited because of the object directly in front of the vehicle and the windowing function has identified the NS, slight left turn, set as being the most activated. The defuzzification routine will therefore only consider the activation of the NM, NS and ZE sets. This yields a defuzzified output which will cause the desired left turn around the obstacle.

5.7.2 Rule Spreading

The second problem with using inhibitive rules is that there may be no active output sets if the obstacle avoidance routine has inhibited all the activated sets. To ensure there will be a control output if there are any output sets which could be used the process of *rule spreading* activates all the output sets to some degree so there are no sets with zero activation.

Rule spreading is similar to declaring the output sets to be bell functions which have non-zero membership over the whole universe of discourse. It operates by using a bell function to add activation values to all the entries in the fuzzy fit vector depending on their distance from activated sets. The activation of the r'th set is multiplied by $e^{-k(i-r)^2}$ and added to the activation of the i'th set. The constant k is used to define the shape of the bell function. The effect of rule spreading on a fuzzy fit vector with just one active set can be seen in Figure 5.11.

Rule spreading tends to modify the final output by biasing it towards the centre set of

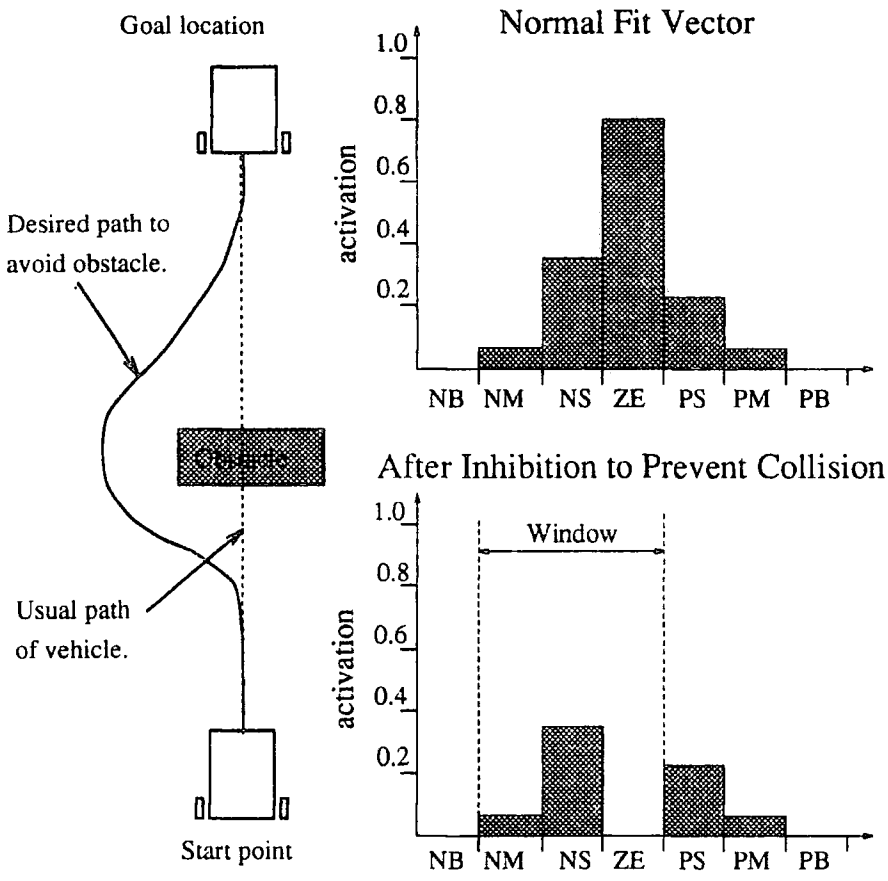

Figure 5.10: Fuzzy Fit Vector Before and After Inhibition to Avoid a Possible Collision, with Windowing.

the output fit vector. This is because an activated set at one end of the fit vector will add some activation to the central sets but since there are no sets to activate further from the centre the defuzzified output value is shifted towards the centre. When used with the AGV controller this has the effect of reducing the steering angle that is applied. It is difficult to decide upon an optimum value of the spreading constant k but reasonable performance can be obtained with values of $log_e 2$ or $log_e 4$ which add 1/2 or 1/4 of the influence of a set to its immediate neighbours respectively. A further examination of the effects of the spreading constant can be found in Section 6.4.

5.8 Windowing and Rule Spreading Combined

When the two methods are combined windowing ensures that unwanted outputs cannot occur while the rule spreading ensures that all output sets are activated to some degree to provide sensible alternative outputs and that activated sets outside the defuzzification

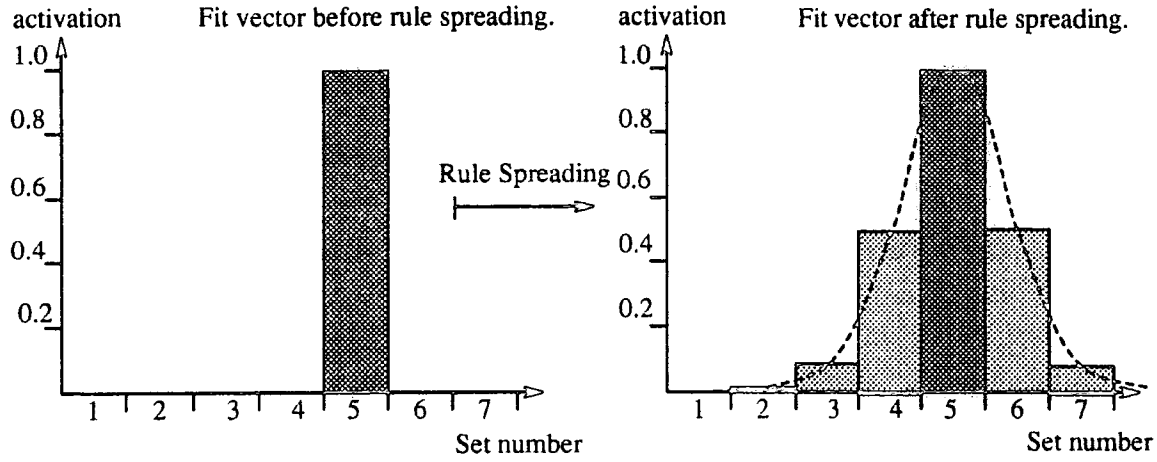

Figure 5.11: Rule Spreading Applied to a Fuzzy Fit Vector.

window still have some influence on the final output.

The navigation and obstacle avoidance functions are combined by multiplying the spread fuzzy fit vector from the navigation function by the mask generated by the obstacle avoidance function. Since the output of the navigation rules is spread over all the output sets there is always at least one activated set which is not masked, unless a collision is inevitable. The resultant vector is then passed through the windowing routine. This selects the output set with highest activation and passes only that set and its immediate neighbours to the defuzzification stage. Defuzzification by the centre of gravity method is applied and an output steering angle is produced. A block diagram of the controller is shown in Fig. 5.12.

5.9 Effects of the Obstacle Avoidance Controller

As discussed in Section 5.1 an ideal controller would be able to guarantee obstacle avoidance without having any unnecessary effect on the normal action of the navigation controller. As has already been seen in Figures 4.11 to 4.14 the introduction of windowing and rule spreading do affect the navigation controller performance, even if only slightly. Their introduction does however also give a beneficial effect in removing the indecision problems described in Section 4.2 where two activated rules at opposite ends of the fuzzy fit vector produce a resultant output close to the centre of the output universe of discourse (often zero). This is illustrated in Figure 5.13 which shows the simulated path of a vehicle guided by a fuzzy logic controller from $(8.7, 7.0)$ to a goal position at $(7.0, 7.0)$ with both the start and goal orientations parallel with the Y axis. The start point has been selected to be right

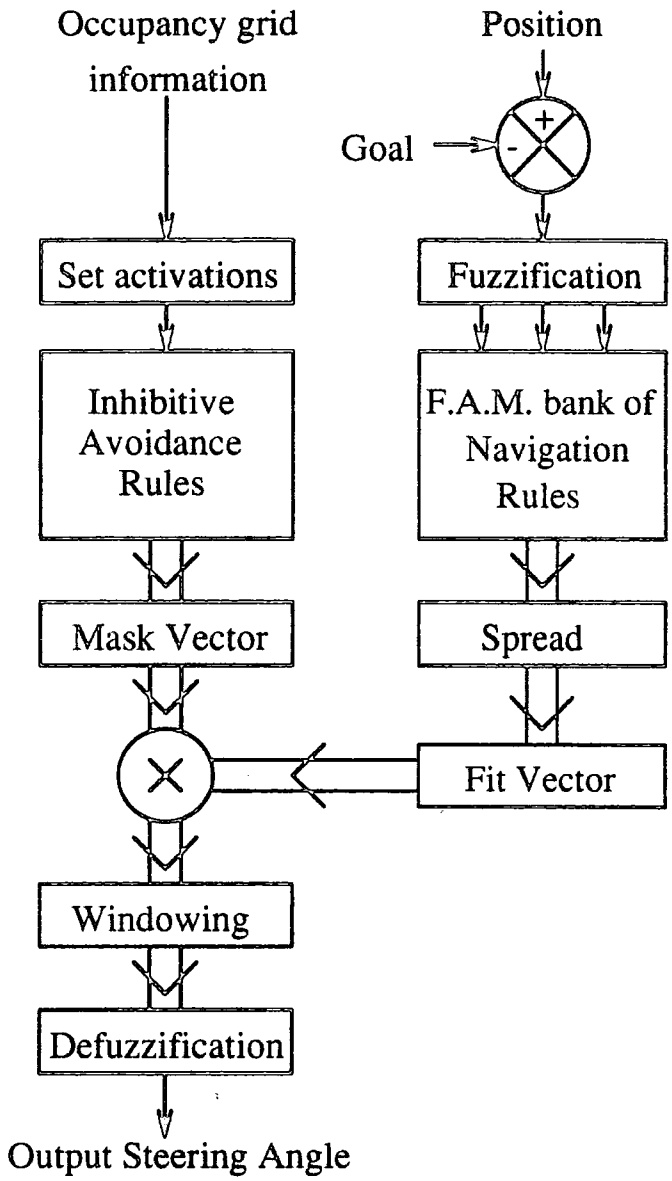

Figure 5.12: Block Diagram of Controller

at the boundary point between a decision to turn sharp left towards the goal or sharp right to approach it from below. The effect of the window in preventing the initial indecision can be clearly seen.

The performance of the full controller in the presence of obstacles can be seen in Figures 5.14 and 5.15. From two different starting positions the vehicle has to negotiate the same obstacle, a small box behind the goal position. The data for these tests was obtained in the laboratory using the research vehicle rather than from the simulation program as has been the case for the previous results. These results clearly show the vehicle successfully avoiding the obstacle with a reasonable amount of clearance between the vehicle and the

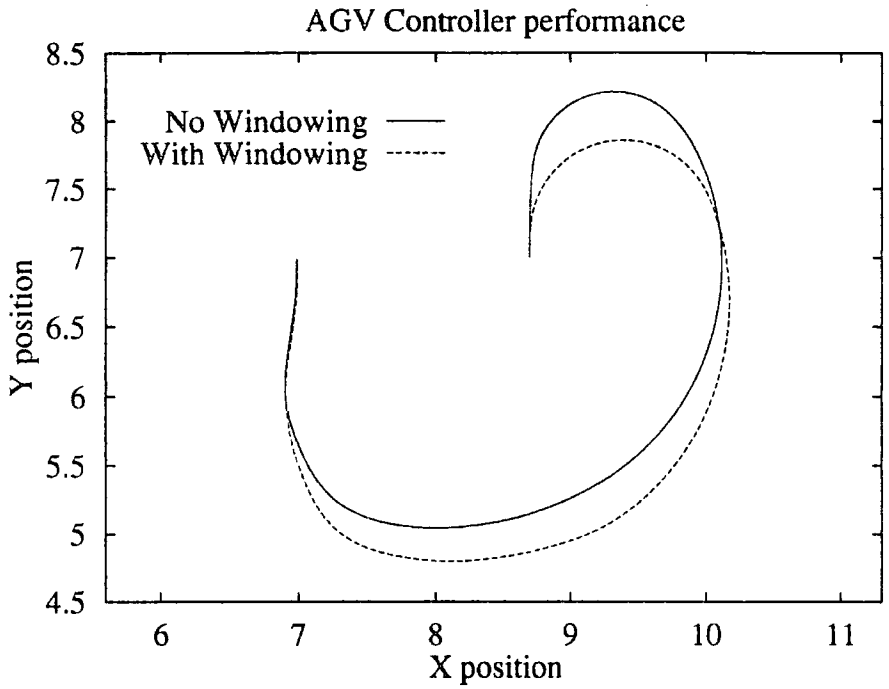

Figure 5.13: Windowing Alleviating a Conflict in the Rules.

obstacle and the vehicle successfully navigating to the goal.

A comparison of the real and simulated paths of the vehicle for a more complex test are shown in Figure 5.16. The vehicle has to negotiate a gap between two obstacles before it can reach the goal and it can be seen that there is good correlation between the simulated results and those obtained from the vehicle.

5.10 The Controller in Action.

To explain in more detail the way in which the navigation and obstacle avoidance controllers act to guide the vehicle this section examines the working of the controllers at several different stages during the test shown in Figure 5.16. The positions are shown in Figure 5.17 labelled from one to four with the obstacles being named walls A,B and C.

The vehicle starts at position 1 which is $(2, 1)$ with orientation 90° and the goal is at $(4, 5)$ orientation 0°. Applying Equations 4.1 to 4.4 described in Chapter 4 this gives state variable values of distance 4.47m, θ_{HE} 63.4° and θ_{GE} 26.6°. Using the set definitions shown in Figure 4.5 distance is therefore entirely a member of the set Large, to degree 0.91. From

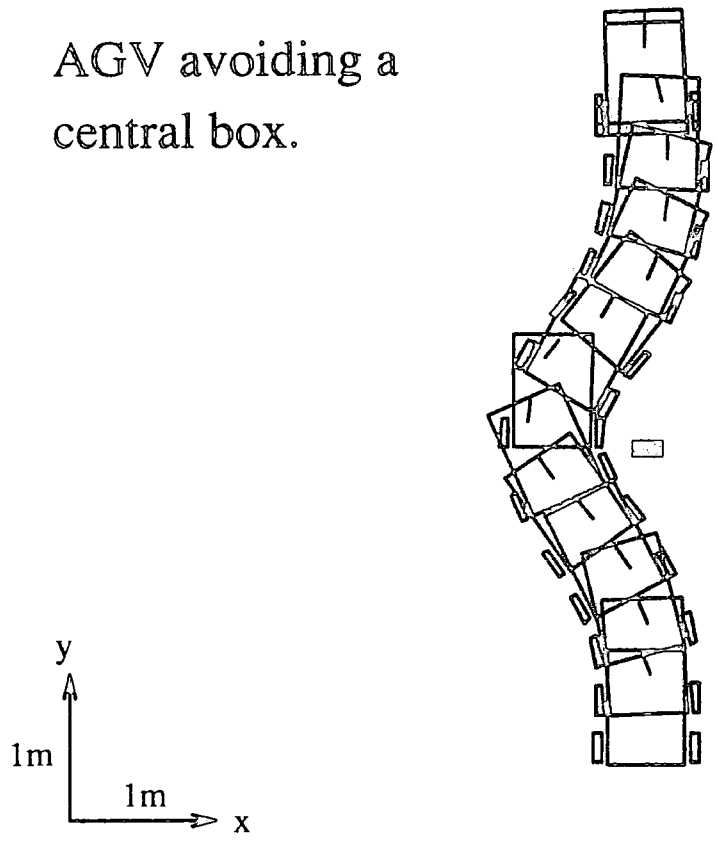

Figure 5.14: Vehicle in Laboratory Test Avoiding a Central Box.

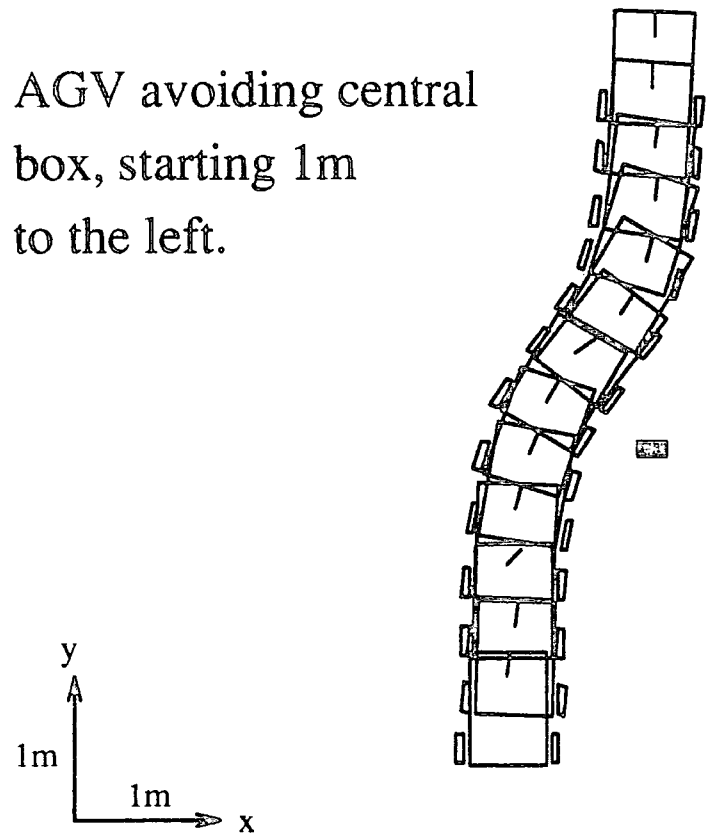

Figure 5.15: Vehicle in Laboratory Test Avoiding a Central Box, Starting 1m to Left.

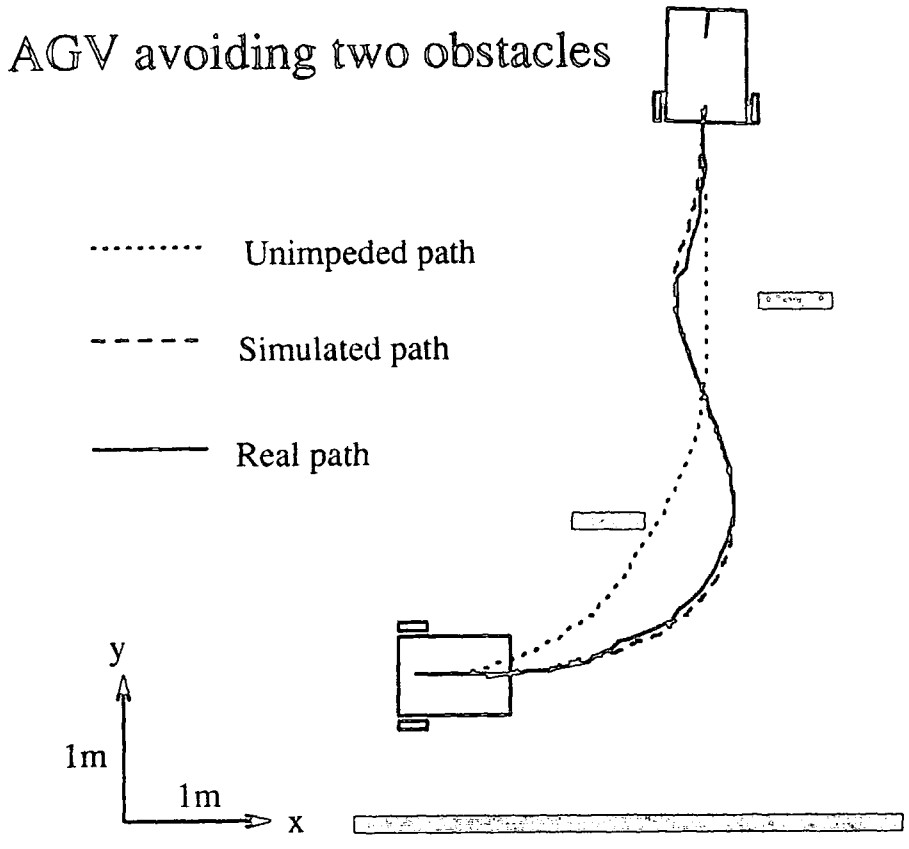

Figure 5.16: Vehicle Avoiding Two Obstacles.

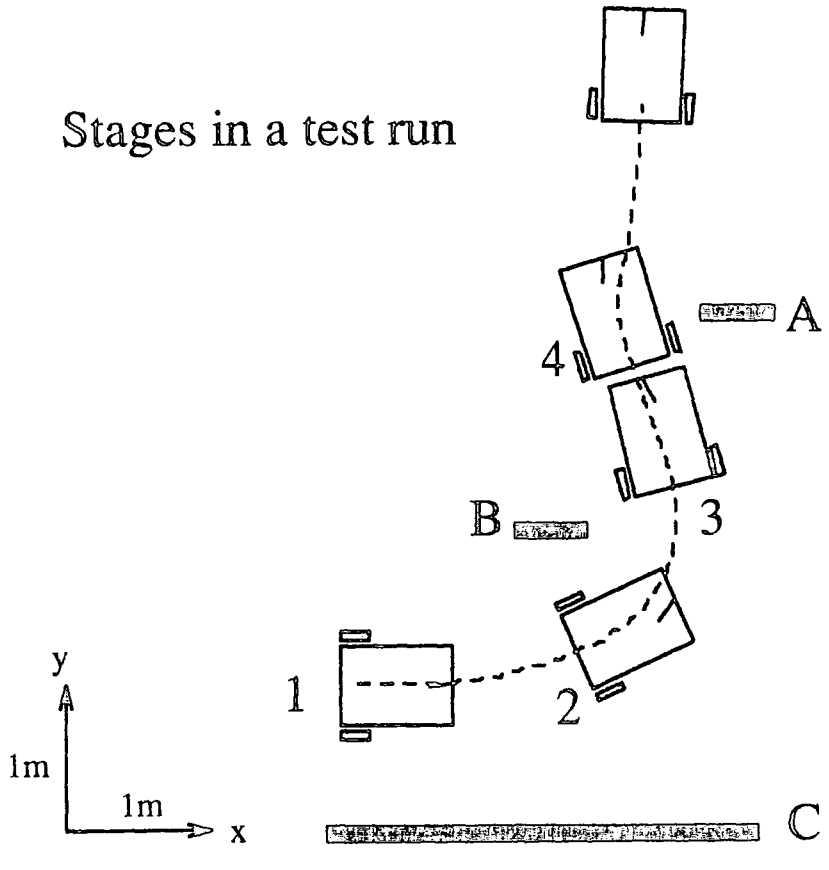

Figure 5.17: Positions used for Analysis of Controller Behaviour.

Figure 4.7 the heading error can be seen to be a member of two sets, Positive Perpendicular to degree 0.17 and Positive Medium to degree 0.26. The goal error is also a member of two sets, Positive Medium to degree 0.43 and Positive Small to degree 0.34. This results in the following four rules (or fuzzy associative memories) being activated.

L, PP, PM $\mapsto$ NB (to degree 0.17)
L, PP, PS $\mapsto$ NB (to degree 0.17)
L, PM, PM $\mapsto$ NS (to degree 0.26)
L, PM, PS $\mapsto$ NS (to degree 0.26)

The rules are displayed in a shorthand format. The first rule should be read as:-

> If the distance to the goal is Large and the heading error is Positive Perpendicular and the goal error is Positive Medium then steer Negative Big.

The resultant fuzzy fit vector can be seen in Figure 5.18 where the activation of the NB and NS steering sets can be seen. The effect of rule spreading is also shown. This gives a large activation to NM as it adjoins both activated sets and some activation to other sets. The activation given to the PM and PB sets are too small to be seen on the same scale. The mask vector shows the effects of the obstacles, wall B has totally inhibited the activation of the NB and NM sets and partially inhibited the activation of the NS set. Wall C has totally inhibited the PB and PM sets and partially inhibited the PS set. The resulting combination yields a combined fit vector with the maximum activation given to the ZE set and so this is where the window is centered. This results in a defuzzified output of $-8°$ resulting in a slight left turn.

By the time the vehicle reaches position 2 which is $(3.63, 1.26)$ with orientation $65.7°$. The state variable values are distance 3.76m, θ_{HE} $60.0°$ and θ_{GE} $5.7°$ activating these rules :-

L, PM, PZ $\mapsto$ NS
L, PM, ZE $\mapsto$ NB
M, PM, PZ $\mapsto$ NB
M, PM, ZE $\mapsto$ NB

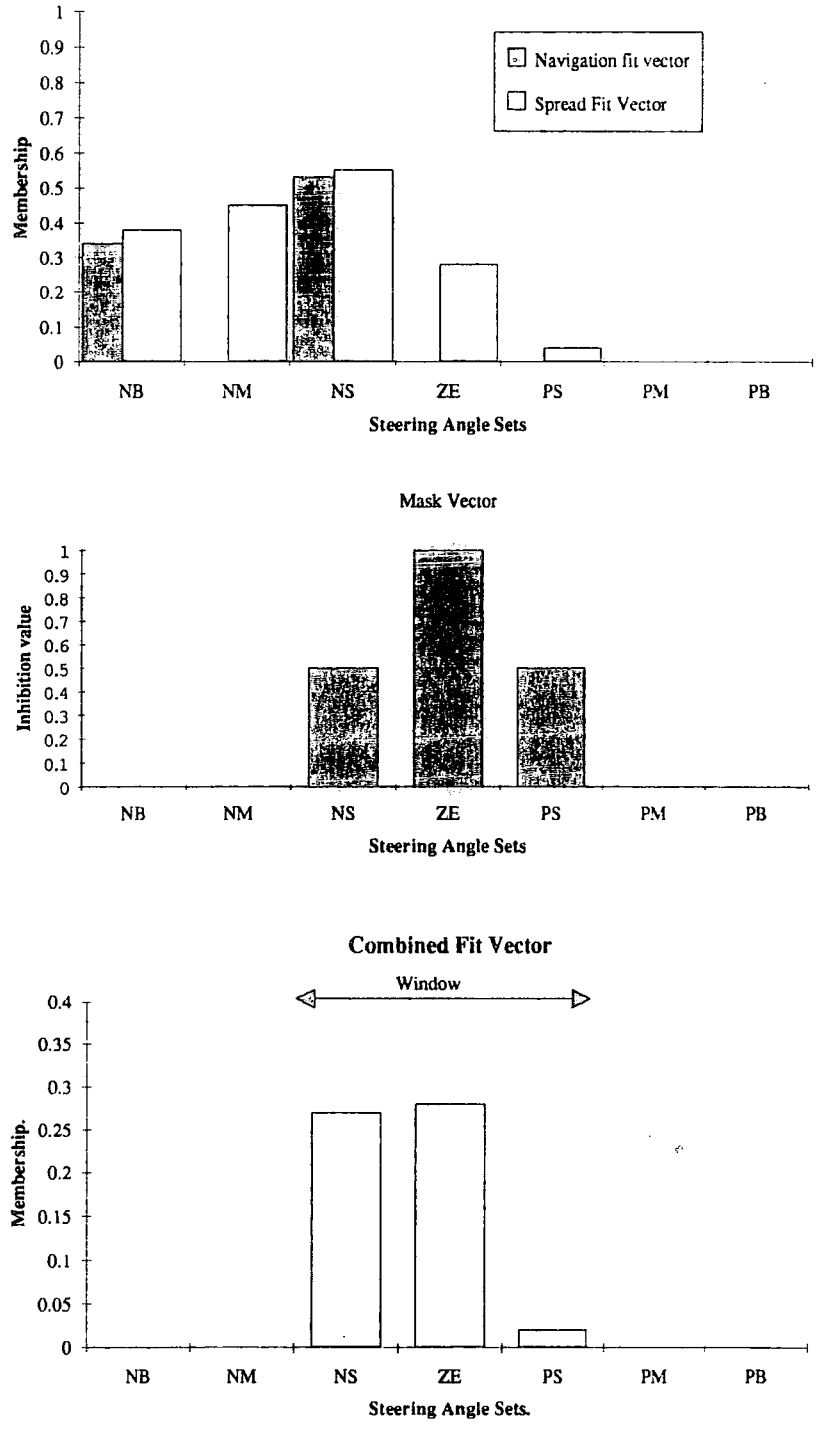

Figure 5.18: Fuzzy Fit and Mask Vectors for Position 1.

As can be seen in Figure 5.19 wall B is still totally inhibiting the NB steering angle set and the presence of wall A is discouraging NS and NM but the combined fit vector has the most activation for steering NM so the sliding window defuzzification gives a steering angle of $-25.93°$ and the vehicle starts to turn around the end of wall B.

The vehicle continues to turn around the end of wall B and reaches position 3, at $(4.19, 2.46)$ with orientation $-13.65°$. The vehicle is now 2.55m from the goal with only small errors in the other state variables θ_{HE} being $-9.3°$ and θ_{GE} $-4.3°$. Consequently only two rules are activated and they are intended to introduce a right turn to line the vehicle up behind the goal position.

M, NS, ZE $\mapsto$ PS
M, NS, NZ $\mapsto$ ZE

Figure 5.20 shows how these rules activate the fuzzy fit vector and also shows that the proximity of wall A is totally inhibiting the output sets ZE to PB inclusive. The only remaining activation therefore is that which has been spread into the NS, NM and NB steering sets. The window is therefore centered on the NS steering angle set and the resulting defuzzified output is $-19.72°$ which causes the continuation of the left hand turn until the vehicle is clear of wall A.

By the time the vehicle has reached position 4, at (3.90, 3.29) it has an orientation of $-13.65°$ which translates to state variable values of distance 1.71m, θ_{HE} $-20.5°$ and θ_{GE} $3.4°$ The vehicle is almost entirely clear of the final obstacle, wall A, and needs to realign itself behind the goal. The following four rules are active in this position:-

M, NS, PZ $\mapsto$ PM
M, NS, ZE $\mapsto$ PS
S, NS, PZ $\mapsto$ PB
S, NS, ZE $\mapsto$ PS

All the rules are instructing the vehicle to turn right so as to curve back towards the goal and the resulting fit vector seen in Figure 5.21 is heavily skewed to the right. Wall A is only a danger to extreme right hand turns, so only the PB output steering set is inhibited. The resulting window, centered on the PM set gives a defuzzified output of 20.5°. The vehicle can now complete its manoeuvres to reach the final goal position and orientation.

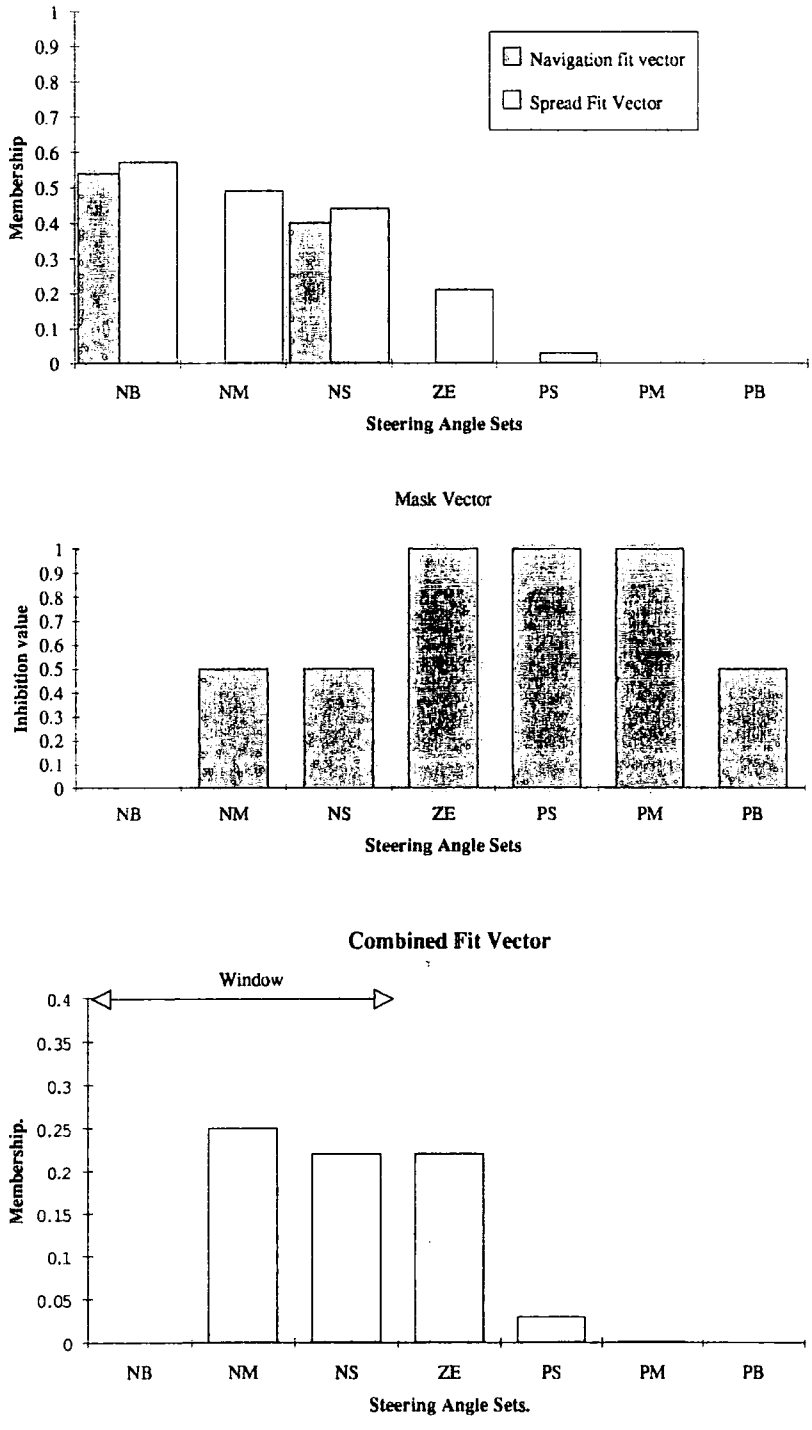

Figure 5.19: Fuzzy Fit and Mask Vectors for Position 2.

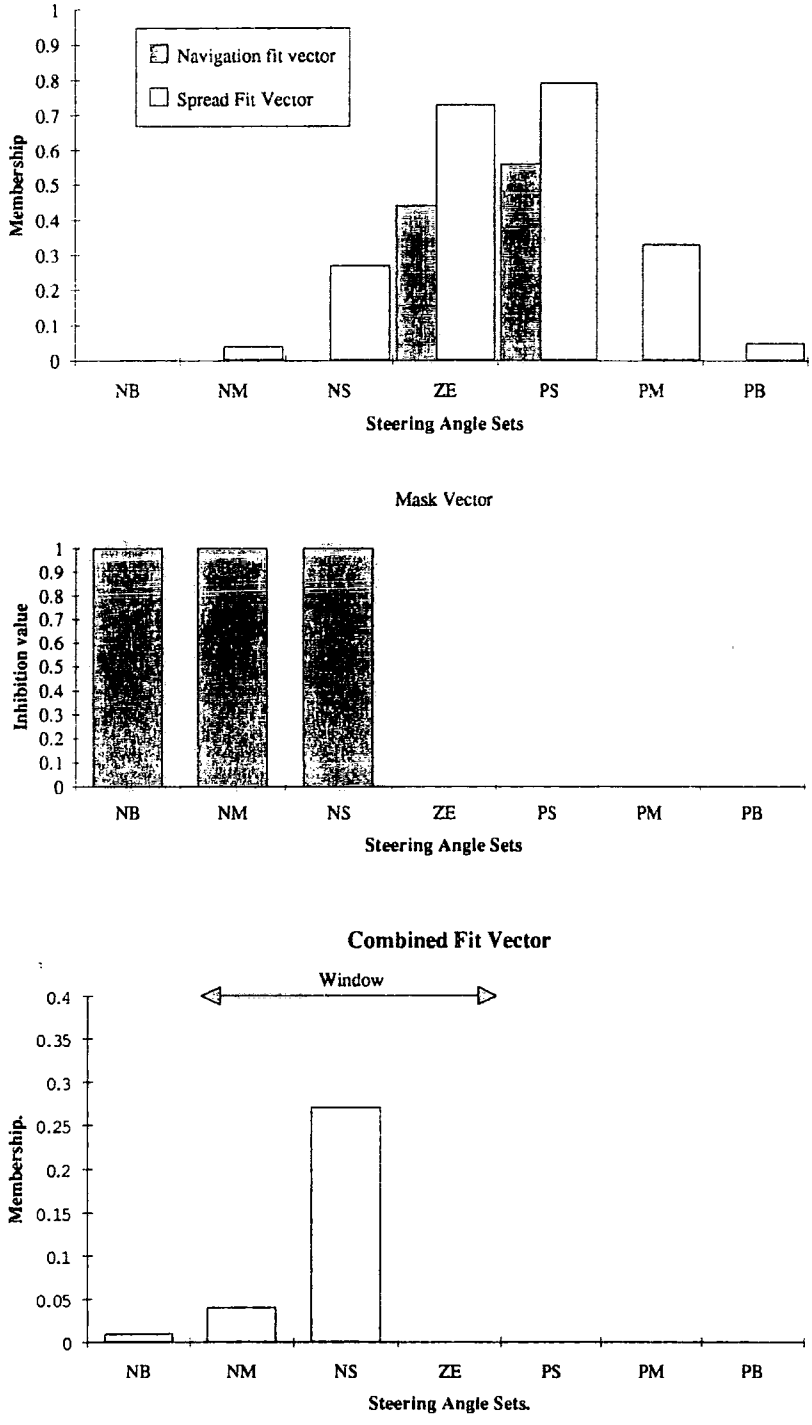

Figure 5.20: Fuzzy Fit and Mask Vectors for Position 3.

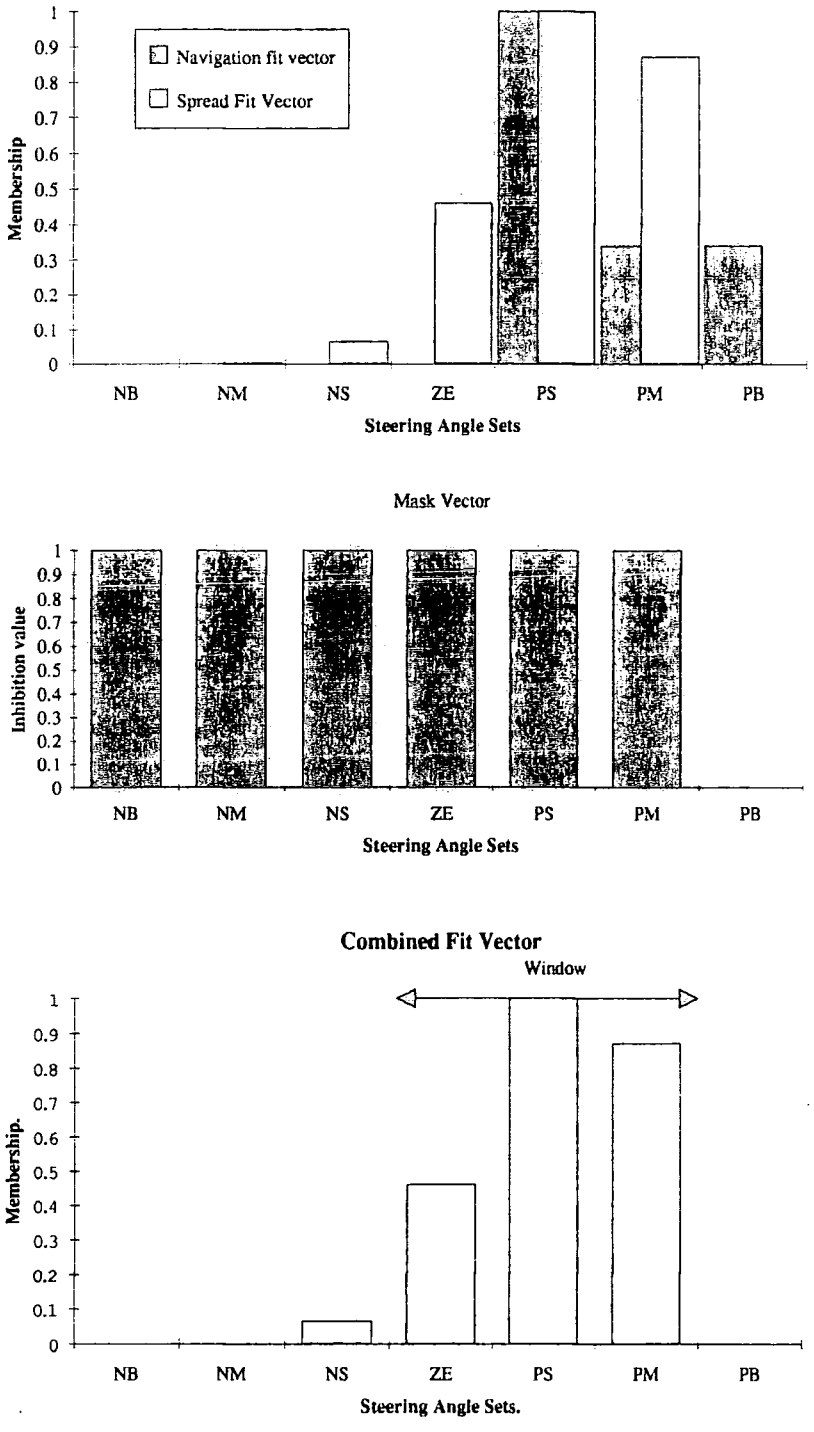

Figure 5.21: Fuzzy Fit and Mask Vectors for Position 4.

Chapter 6

Analysis and Development of the Avoidance Scheme

While the methods outlined in the previous chapters have been shown to give a simple and effective guidance method there are some problem situations which can cause undesirable behaviour. This chapter introduces some of these problems and describes how the controller has been developed to eliminate them. The obstacle avoidance controller described in the previous chapter includes many original features and settings, the values of which were determined empirically. This chapter contains analysis of the effects that changing the values of the rule spreading constant and window width have on the performance of the controller. A method of generating obstacle avoidance sets linked to the geometry of the vehicle is introduced and used to evaluate the effect of altering the dimensions of the avoidance sets and their associated inhibition values. Results are then presented from tests carried out on the vehicle using the settings shown to be most effective by the results of the tests done in simulation.

6.1 Dynamic Indecision

The *windowing* method introduced in the previous chapter provides a solution for the problem of static indecision when the controller is unable to decide between alternative steering angles. There does remain however the problem of dynamic indecision. This is generally

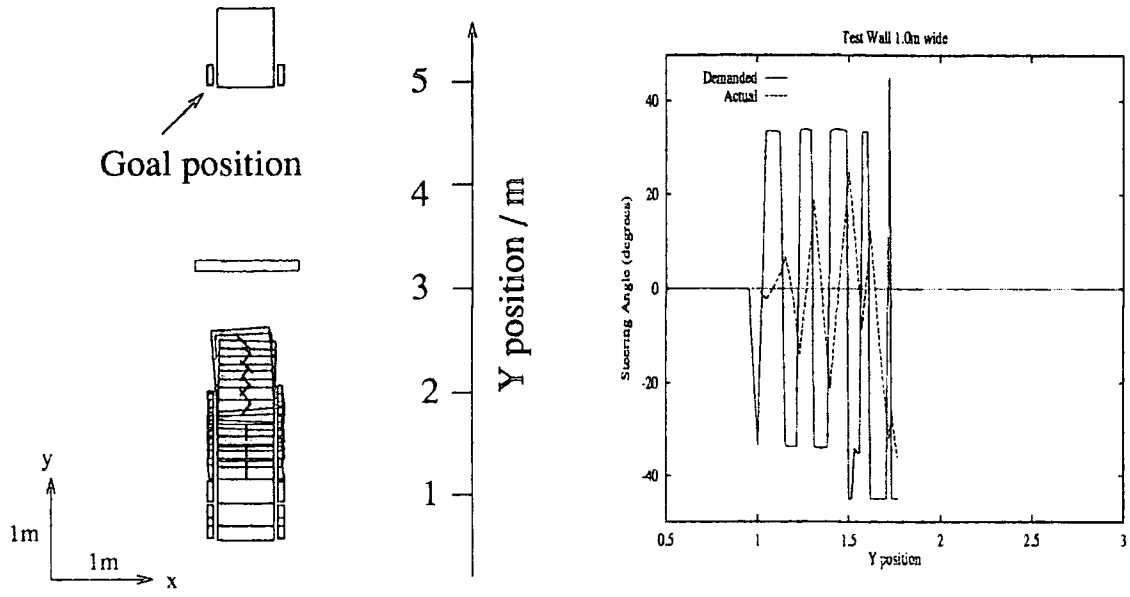

Figure 6.1: Example of Dynamic Indecision Producing Oscillations

found in the case where an obstacle symmetrically blocks the path to the goal so there is no obvious direction to take. A particularly bad situation is the one which is illustrated in Figure 6.1 with a straight path to the goal blocked by a large symmetric wall. The controller recognises that it cannot steer straight ahead but as can be seen from the accompanying graph it oscillates between trying to steer hard left and hard right and the steered wheel never deviates enough from zero to avoid the obstacle. The problem is that as the vehicle is turned away from the wall the navigation controller attempts to turn it back onto the centre line of the goal thus making it more attractive to turn in the opposite direction. Since this is possible the obstacle avoidance system allows it and the vehicle is commanded to steer in the opposite direction. The oscillations are brought about by the conflicting goals of the navigation and obstacle avoidance system and the damping effect of the vehicle dynamics prevents adequate avoiding action being taken. This effect occurs only rarely since if the situation lacks symmetry or the obstacle is small the vehicle can extract itself from the oscillation situation. However with the one metre wide wall shown in Figure 6.1 the oscillation condition leads to a failure. This test was from (4.5, 0.5) to (4, 5) with starting and goal orientations parallel to the y axis. The obstacle is a 1m by 0.1m wall centered about (4, 3).

Examination of the controller in action sheds further light on this problem. Firstly the controller does not take into account the steering dynamics so it fails to account for the fact that a large change in steering angle cannot be achieved instantaneously and therefore fails

to realise that avoiding action which involves small steering angle changes is preferable. In addition to this once a decision has been made to steer the vehicle in a particular direction to avoid an obstacle the controller does not consider this decision when evaluating the situation on subsequent control interrupts. In effect the controller attaches no importance to the previous decision so if it previously decided to go to the left of an obstacle even the smallest bias to the right may cause it to reverse that decision. Clearly there are situations where two equally favourable options are available and it is far better to select one and stick to it than to oscillate between the two possibilities in a state of dynamic indecision.

To reduce the effects of these problem two new rules were introduced, one to cause the vehicle to favour the present actual steering angle as an output to take account of the time taken to implement large steering changes and one to make the controller favour sticking to previous decisions in uncertain cases.

The first rule operates by finding the output set which the current steering angle is a member of and adding activation to it. The level of activation to be added was selected as being 0.1, a tenth of the maximum possible value. This is enough to cause the vehicle to favour the present steering angle while not being too unwilling to change direction. The second new rule adds an activation level of 0.1 to the output set which has the highest activation on the previous control interrupt. This causes the controller to stick to its previous decision unless an alternative output is significantly more favourable rather than altering the steering angle drastically to a setting only marginally more favourable than the present one.

The rules might be described as:

PREFER The present steering angle.
and
PREFER The previous demanded steering angle.

Where PREFER indicates a rule with a fit value of 0.1. Clearly this value was derived empirically as giving a good compromise between higher values which might affect the navigation controller too drastically, delaying necessary changes in steering angle, and lower values which might be unable to satisfactorily prevent dynamic indecision. Since these effects also depend upon the values used throughout the controller for rule spreading, inhibition and windowing it was not considered important or practical to make an exhaustive

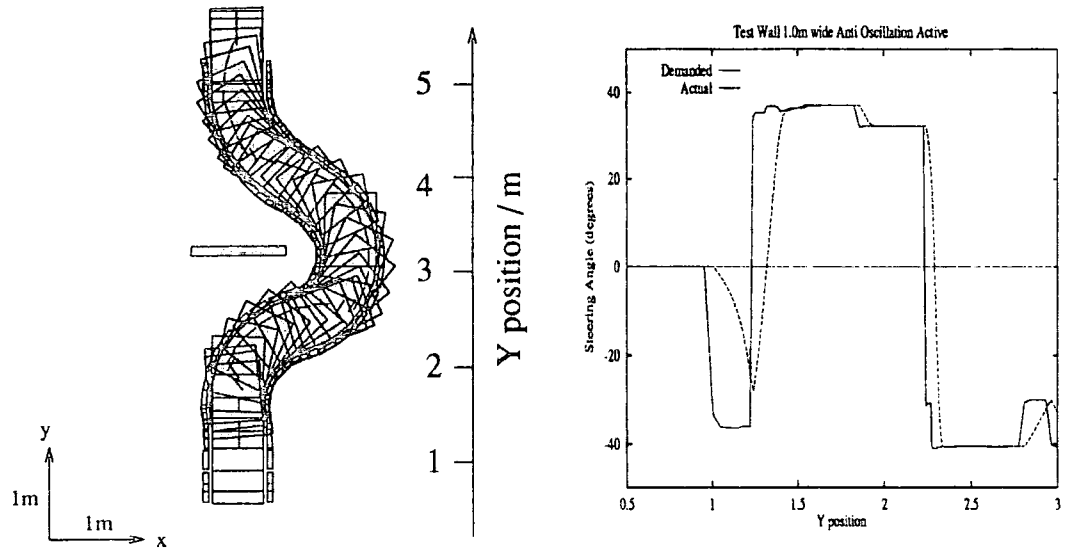

Figure 6.2: New Rules Preventing Dynamic Indecision

study to try and optimise these values.

The effect of these rules can be seen in Figure 6.2 which shows the same situation as in Figure 6.1 but the new rules have removed the dynamic indecision and the controller successfully guides the vehicle around the obstacle. A single oscillation is present as the vehicle initially tries to go to the left of the obstacle before changing to go to the right but the situation is much improved. Figure 6.3 shows the vehicle successfully avoiding a two metre long wall directly between it and the goal demonstrating that the new rules have greatly improved the performance of the controller in these situations.

6.2 Demand Velocity Oscillations

The previously mentioned oscillations in demand steering angle have another detrimental effect. The sudden large changes in steering angle lead to large errors between the actual and the demanded steering angle which in turn cause a reduction in velocity to be better able to meet these sudden changes. Large changes in demand steering angle are inevitable when there are obstacles present since the compromise steering angles between two values may be inhibited to avoid collisions. If obstacles are not present however the changeover from one steering direction to another can still happen quite fast and cause an unnecessary dip in velocity as a large difference between the actual and demanded steering angles builds up.

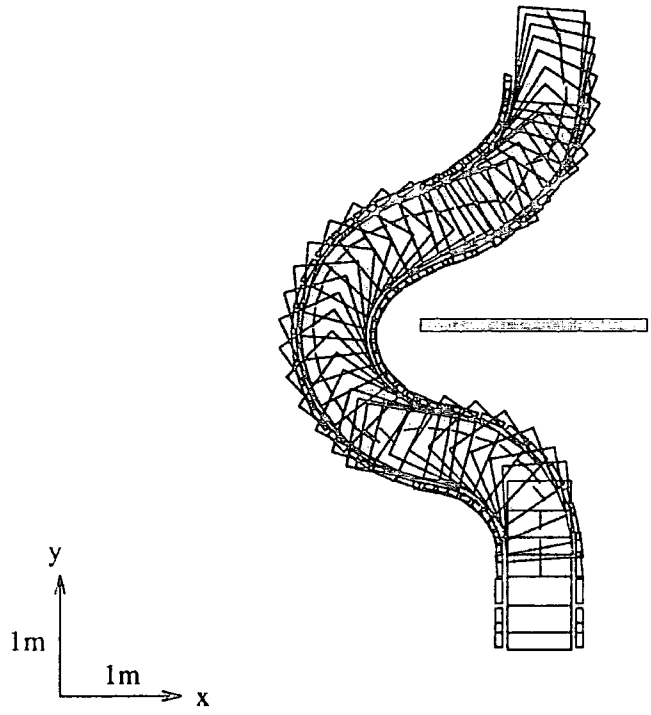

Figure 6.3: New Rules Navigating Around a 2m Long Wall

This is illustrated in Figure 6.4 where the simulated vehicle is travelling from (3, 1) to (4, 4) with the start and goal orientations parallel to the Y axis. As the right hand graph shows there are sharp changes in demand steering angle which in turn produce dips in the vehicle velocity.

The sharp change in demanded steering angle is caused by the need to change from a right hand turn to approach the extended center line of the goal to a left hand turn to line up with the goal orientation. The fuzzy navigation controller reduces the activation of the right hand steering sets and increases the activation of the left hand steering sets as this changeover point approaches. However, the windowing technique effectively masks this gradual change and the demand changes dramatically the instant the activation of the left hand steering sets is slightly larger than that of the right hand steering sets. In this case the compromise between going left or right should be taken as being close to zero since it represents a smooth transition rather than indecision.

The problem therefore is to distinguish between the cases when two heavily activated but distant output sets represent indecision on the part of the controller and where they represent a transitional phase between two sections of the vehicle's path. It is desirable to extend the width of the window from the fixed setting, which includes only the maximum

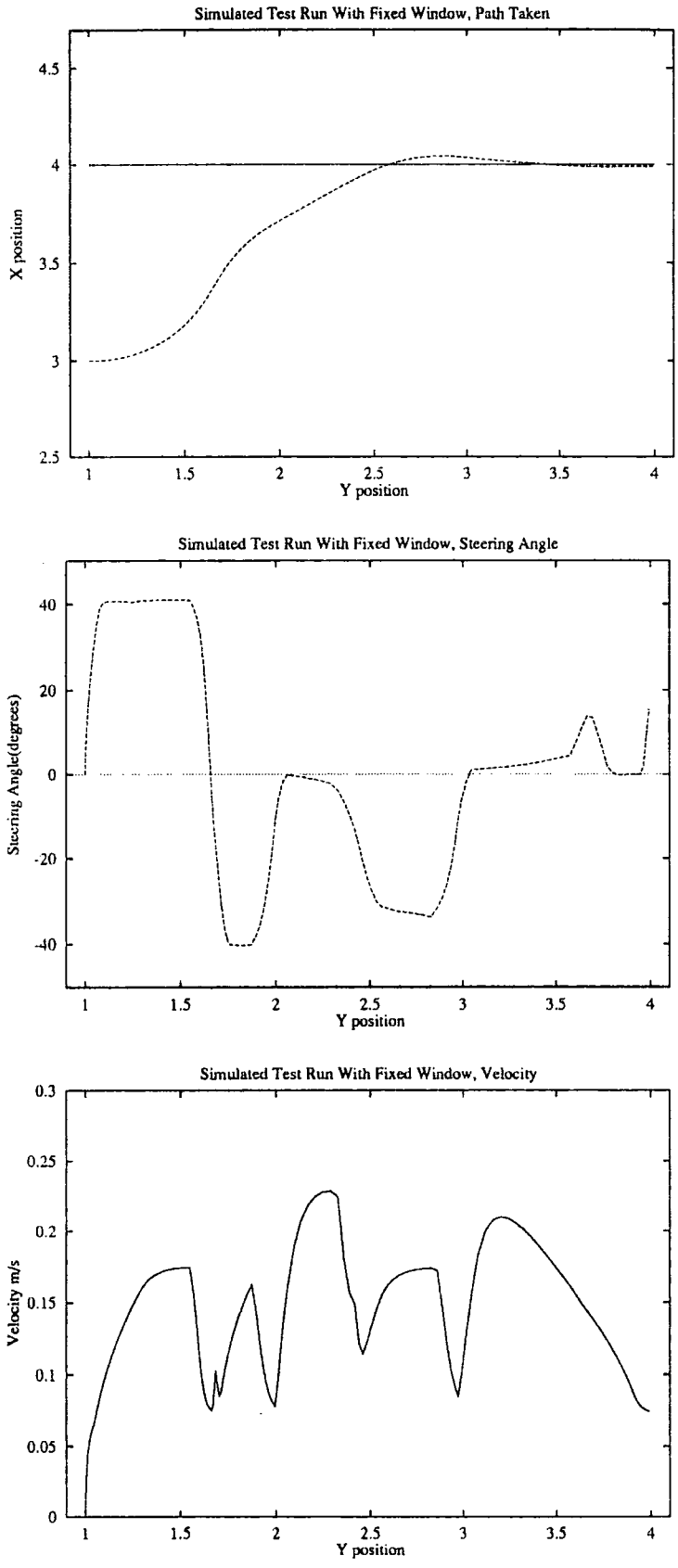

Figure 6.4: Test Illustrating Sharp Changes in Demand Steering Angle

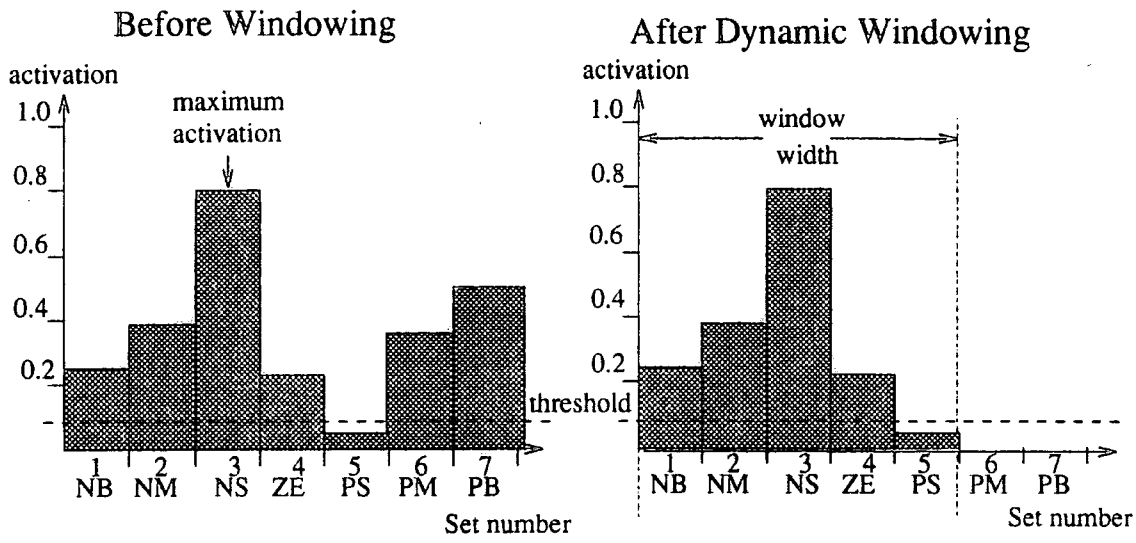

Figure 6.5: An Example of Dynamic Windowing

activated set and its two neighbours, to achieve smoother transitions, but not so far as to reintroduce indecision.

6.3 Dynamic Windowing

The method developed to do this has been called *dynamic windowing* and is a modification of the windowing technique described in Section 5.7. Using dynamic windowing the size of the window is expanded out up to a maximum window size (set at four) so long as the activation of the included sets exceeds a threshold value (set at 0.1). The window size is initially set to be one. If the two neighbours of the output set with the most activation are activated greater than the threshold then the window is expanded to include their neighbours and the process repeats until the maximum window size is reached or a set activated less than the threshold value is found.

An example is shown in Figure 6.5 where the maximum activated set is the NS, Negative Small, set and the PS, Positive Small, set is below the threshold value, perhaps due to a high probability of there being an obstacle in that direction. The window therefore doesn't expand beyond the PS set. The example shown in Figure 6.6 is one where all the values exceed the threshold and there are large activations for the NB and PB steering sets. This could indicate a change from one curve to another as in Figure 6.4 or indicate a conflict in the rules between turning left or right as in Figure 5.13. For this reason the maximum

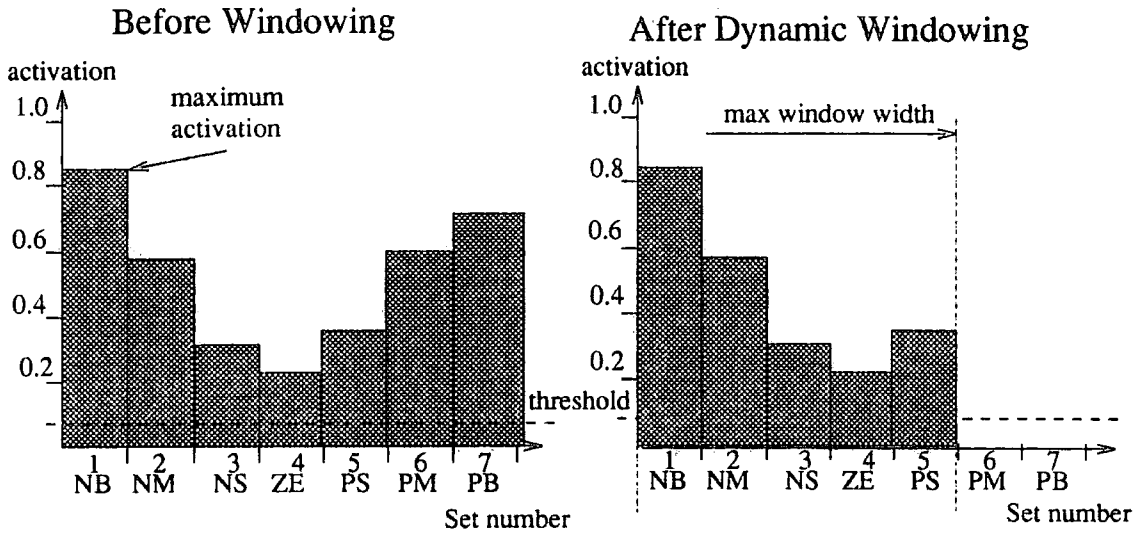

Figure 6.6: A Second Example of Dynamic Windowing

window width has been set at four so that the window does not extend to include the opposite steering extremes. If the situation is one where there is a conflict the suggestion to go in the opposite direction will reduce on subsequent interrupts while if it is necessary it will increase and a gradual changeover will begin.

The effect of using dynamic windowing can be seen in Figure 6.7 which shows the improvements made over the same test which was shown in Figure 6.4. The change in steering angle is less severe, enabling the velocity controller to maintain the vehicle at a higher velocity. As well as giving a smoother trajectory with less acceleration and deceleration this gave a ten percent decrease in the time taken for the test run.

The major problem with dynamic windowing is that it can affect the path taken by the vehicle around an obstacle and also alter the behaviour of the navigation controller. From tests it has been shown that the correct threshold setting depends to a large degree upon the value used for the spreading constant and on the inhibition values used in the definition of obstacle avoidance sets. The following section presents results from a large series of tests using different settings of the controller parameters to investigate their effects and draw conclusions about the effectiveness of different techniques and settings.

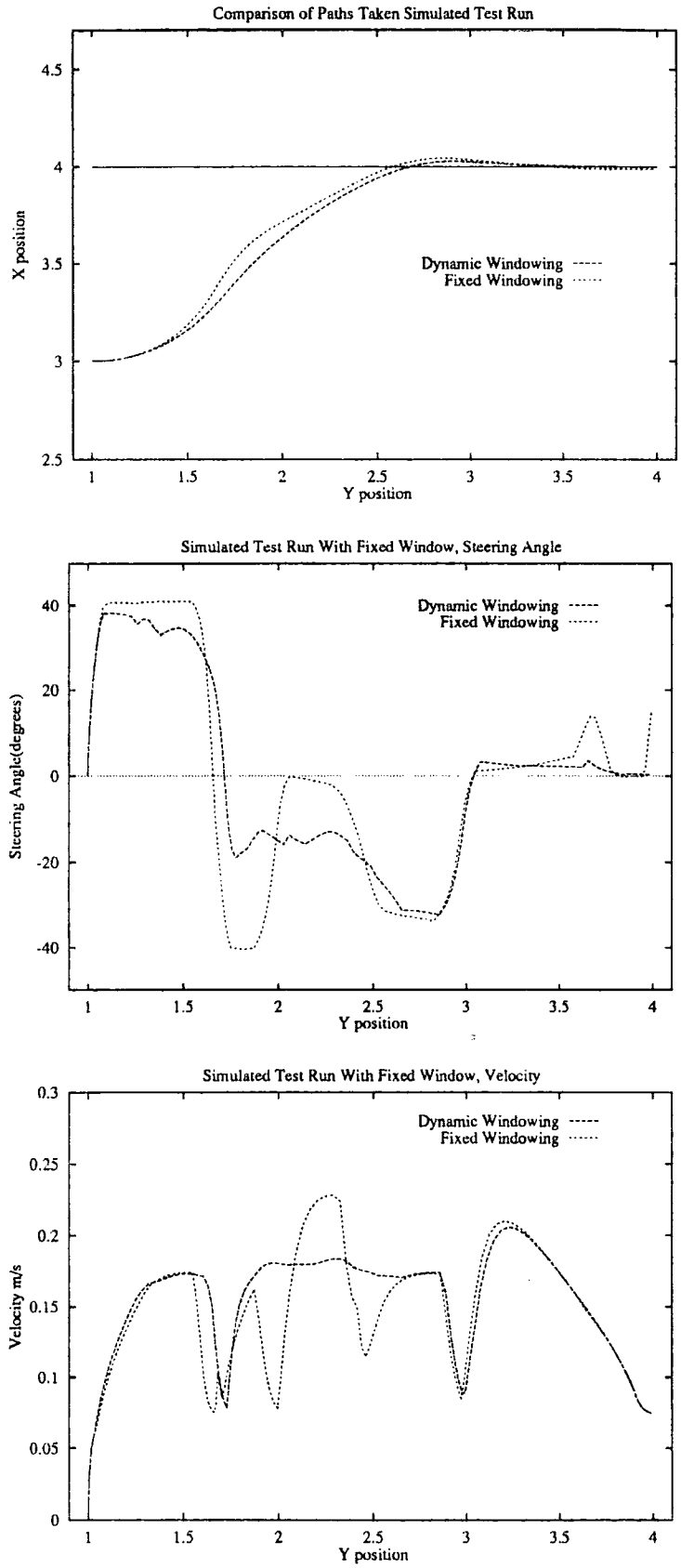

Figure 6.7: Test Illustrating Smoother Transitions With Dynamic Windowing

6.4 Spreading Constant Values

When rule spreading was introduced in Section 5.7 it was stated that the values of the spreading constant k which were used were log(2) and log(4) being 0.7 and 1.39 respectively. These values were chosen because they spread one half and one quarter of the influence of an output set to its immediate neighbours. The value of the spreading constant can be seen as indicating how seriously the controller considers alternatives to the path suggested by the navigation controller. The lower the spreading constant the more influence is spread through the fit vector. In the limit a spreading constant of zero would flatten the fit vector, totally removing the influence of the navigation controller. It would seem unreasonable for values below $log(2)$ (0.7) to be used since this is capable of altering the peak value in the fit vector, as an output set with two highly activated neighbours might then receive more activation than either of its neighbours and become the most activated set. This violates the constraint that the obstacle avoidance method should not significantly alter the performance of the navigation controller. In some cases, however, a low spreading constant (< 0.7) might be necessary to achieve a reasonable level of obstacle avoidance.

To examine these issues and the effect of dynamic windowing a series of tests was performed with values of the spreading constant as 1.39, 0.7 and 0.29 which spread 0.25, 0.5 and 0.75 respectively of the influence of an output set to its immediate neighbours. The results were obtained from the simulation program using improved obstacle set definitions known as *short* which were chosen for their overall level of performance from the early tests on the geometrically defined sets described in Section 6.6.

Figure 6.8 shows the path taken by the vehicle to avoid a 0.6m wide wall and a 2m wide wall with different values of spreading constant. The vehicle is started at (4, 0.5) and heads to a goal at (4, 5) with start and goal orientations of zero degrees. The symmetry this gives to the problem makes it a worst case scenario for the controller. In certain cases the vehicle went left around the obstacle not right. In these cases the path has been reflected around the axis of symmetry in order to allow easy comparison of the paths and is noted in the key for the traces.

In both these tests and in others it can be seen that the weaker spreading constant of 1.39 fails to give enough obstacle avoiding behaviour for the vehicle to safely avoid the obstacle and the run is terminated early. For the smaller wall there is little difference

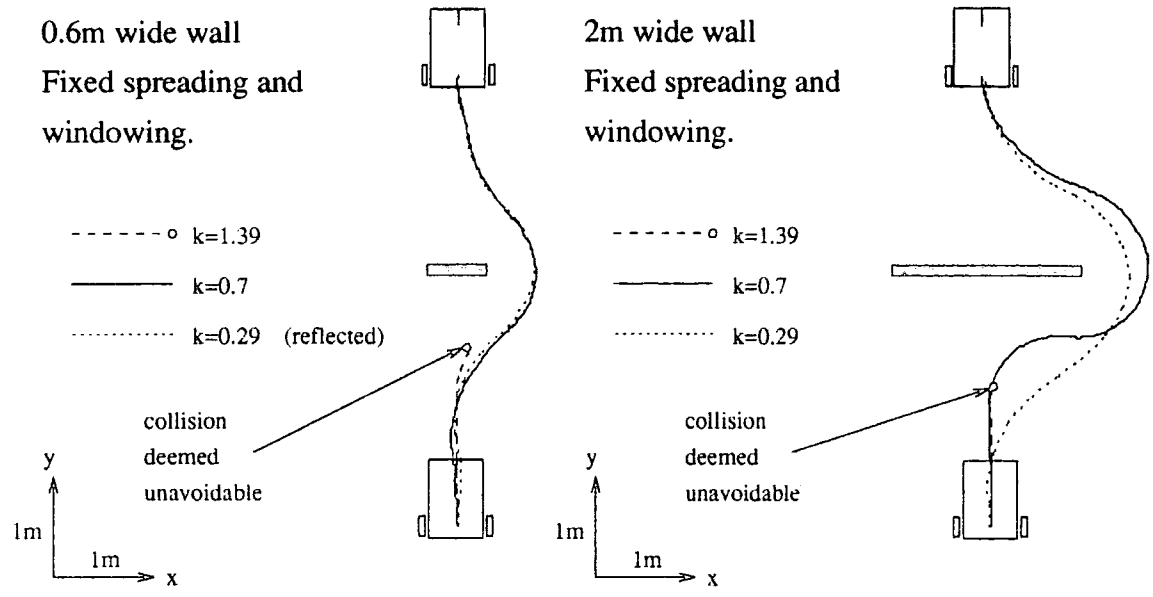

Figure 6.8: The Effect of the Spreading Constant on Obstacle Avoidance

between the paths of the two lower values of spreading constant. However, for the 2m wide wall it can be seen that only the very low value of 0.29 takes early avoiding action while with a spreading constant of 0.7 the vehicle leaves the turn till later and has to take much more extreme avoiding action.

From this it can be seen that where large obstacles, inhibiting more of the potential output sets, are involved a smaller spreading constant gives a more positive avoiding response. For the reasons discussed above, however, it is not desirable to have small spreading constants all the time because of the detrimental effect on the navigation controller in the absence of obstacles. Use of small spreading constants also renders dynamic windowing ineffective since the flattening effect it has on the fit vector reduces the differences between activations determined by the navigation controller and smoothes out the path so much as to make the vehicle fail to reach the goal. This is illustrated in Figure 6.9 where the dotted path of the lower spreading constant wanders past the goal, unable to line up with the goal orientation due to the relative weakness of the navigation rules.

6.5 Variable Rule Spreading

Following on from these results it is obviously beneficial to have a lower spreading constant in the presence of large obstacles but not with smaller obstacles or during navigation through

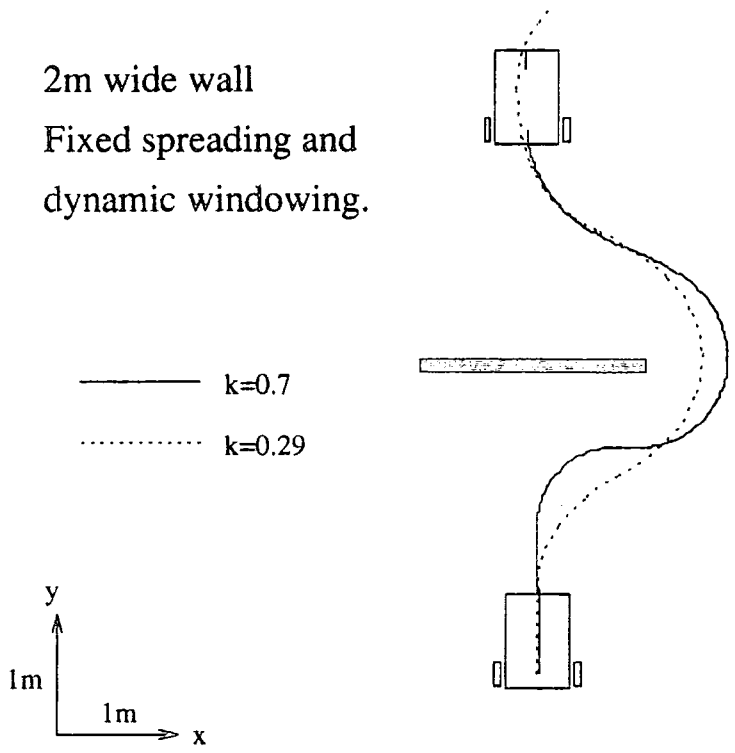

Figure 6.9: The Effect of the Spreading Constant on Obstacle Avoidance with Dynamic Windowing

clear areas. To achieve this it is necessary to have a variable value of the spreading constant. A simple and effective way of achieving this is to have the spreading constant depend on the average level of inhibition present in the mask vector. If there are no nearby obstacles the mask vector has an average value of 1 and it then reduces towards zero as the objects activate more obstacle avoidance sets and the resulting lower spreading constant gives more weight to possible avoidance paths. Following a series of tests it was decided to vary k with the square of the average value in the mask since this increases the spreading more dramatically with larger obstacles. Equation 6.1 calculates the spreading constant k from the entries in the mask vector $M(1..7)$. The maximum spreading is set by the value of k_{max}.

$$k = k_{max}\left(\frac{1}{n}\sum_{i=1}^{n} M[i]\right)^2 \qquad \text{for vector of size } n \tag{6.1}$$

Tests undertaken varying the value of k_{max} show that with the higher value of 1.39 results were improved slightly but the vehicle still failed to navigate around the 2m wide wall. Using k_{max} as 0.7 however the results presented in Figure 6.10 show that the path has been improved around the wider wall and is still good in the smaller test. Furthermore,

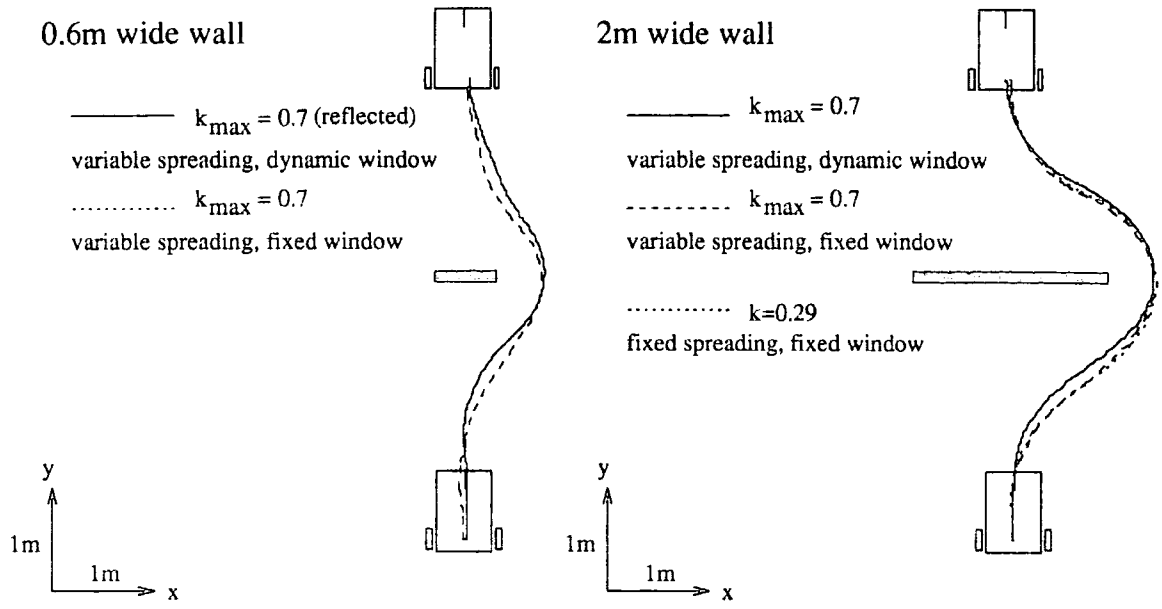

Figure 6.10: The Effect of Variable Spreading and Dynamic Windowing on Obstacle Avoidance

tests incorporating the dynamic windowing technique follow a similar path but exhibit the smoother steering changes, enabling reductions in the time taken of a few percent and avoiding unnecessary braking. The dynamic windowing technique while it does not drastically improve avoidance performance does enable a smoother, faster path, particularly in the open space cases introduced earlier.

In conclusion a series of tests was undertaken in simulation with varying wall sizes and spreading constants, with fixed and variable rule spreading and fixed and dynamic windowing. From these tests, an illustrative subset of which have been presented here, the most satisfactory variation has been shown to be with variable rule spreading using k_{max} at 0.7 and with a dynamic window expanding to a maximum width of four. The next sections move on to address the issues of the correct sizes and inhibition values for the obstacle avoidance sets.

6.6 Geometric Obstacle Avoidance Set Generation

Initially the obstacle avoidance sets were determined arbitrarily and were loosely linked to the vehicle geometry and steering angles as described in Section 5.4. This section introduces the obstacle avoidance generation method used to automate this process. This allows for the possibility of examining the effects of different set definitions and in the longer term the

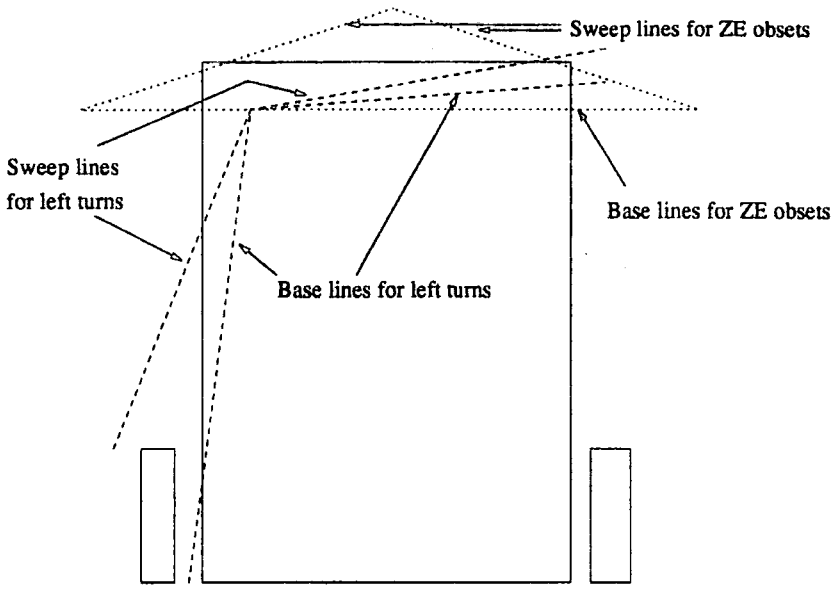

Figure 6.11: Lines Used to Generate Avoidance Set Areas.

generation of sets while the vehicle is in motion, perhaps in response to changes in velocity or obstacle density.

The obstacle avoidance sets are generated by considering the geometry of the vehicle. Lines, known as sweep lines and shown in Figure 6.11, are defined which will sweep out the area the vehicle occupies during a turn. The basic method is that the positions of these lines are defined relative to the vehicle as it starts a turn and then the position the vehicle would be in after travelling a set distance along a curve is calculated. The radius of curvature depends upon the centroid of the output steering angle set to which the avoidance set is to be linked. The new positions of the sweep lines are calculated and the avoidance set areas defined by linking the old and new positions of the sweep lines. The lines linking the start and end positions of the ends of the sweep lines are an approximation to the curve followed by the corner of the vehicle. The shorter the distance travelled and the larger the radius of curvature the closer this approximation is. The exception to this procedure is in the generation of the sets closest to the vehicle when the first points are defined by what are called base lines. These lines are closer in to the vehicle than the sweep lines to take account of the protruding wheels and corners of the vehicle which must be included in the maximum inhibition set closest to the vehicle. Figure 6.12 shows how the avoidance set generation operates for a slight left turn.

The generation program called **avoidgen.c** allows for different base and sweep lines to be used for each steering angle set. Currently, however, all the non-zero steering angle

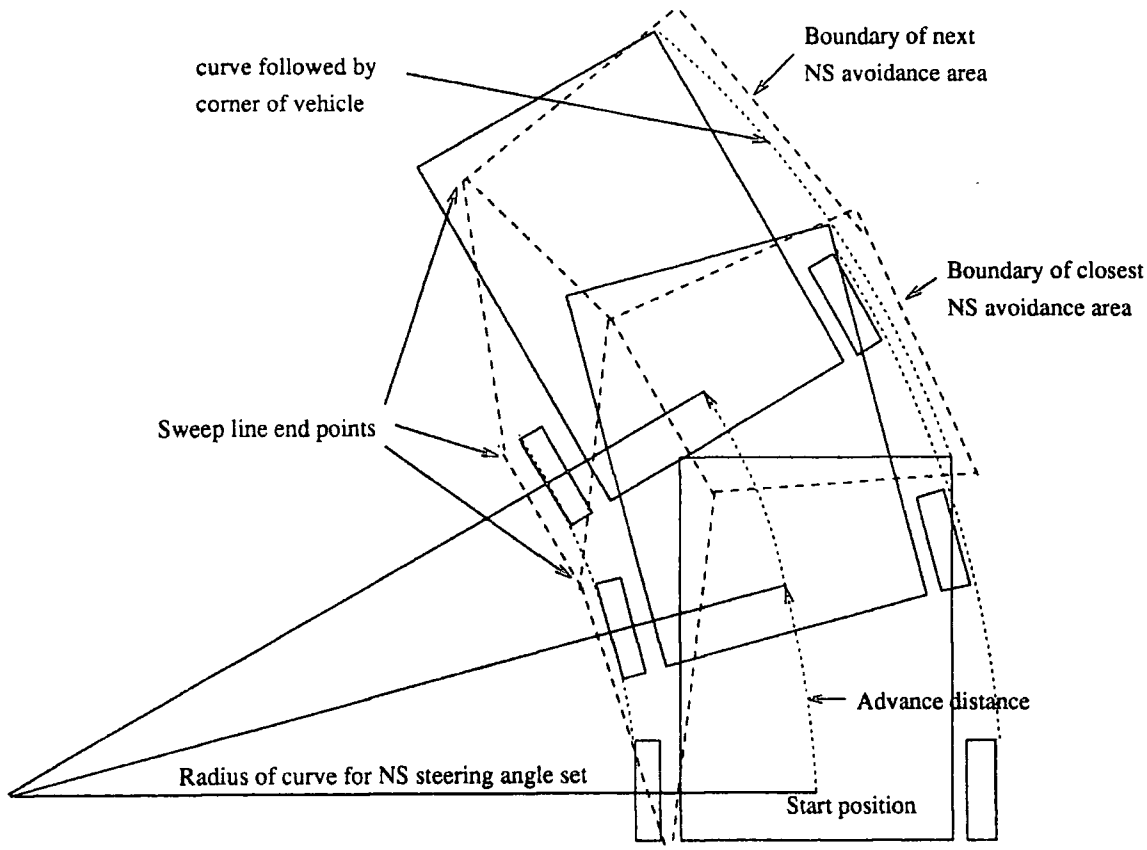

Figure 6.12: Generation of Avoidance Sets for a Negative Small Steering Angle.

sets use the same definitions. Different definitions are used for the zero steering angle sets since only the front of the vehicle is vulnerable to collisions. The zero steering sweep lines are angled to a point at the centre of the vehicle so that the most potentially dangerous objects, those in the middle of the vehicle's path, are detected first. There is a special pair of sets linked to the zero steering angle set which covers rectangular areas in front of the protruding rear wheels to make sure that the controller allows for them. To allow for the fact that the vehicle may steer on a number of possible angles there is provision to widen the area covered by an avoidance set by allowing the ends of the sweep lines to steer on different angles. The zero steering angle set, for instance, might consider the left hand end of the sweep line moving according to a vehicle with a steering angle of $-5°$ and the right hand point on $5°$ to give a wider coverage in front of the vehicle. The effect this divergence has on set definitions is illustrated in Figure 6.18.

For the tests which follow the same total inhibition sets were used in all cases. These sets are easy to define since they are close to the vehicle. The zero steering angle set must be long enough so that it covers the maximum turning radius of the vehicle, so that the vehicle will be prevented from going forward if doing so would bring it so close to an obstacle that even the tightest possible turn would be unable to prevent a collision. This is a distance of 0.7m and it is this distance which is used for the advance distance of the zero and small steering angle sets. The tighter steering angle sets, NM and NB, are a slightly different case. Firstly it is necessary to define the sets as two smaller ones so that the straight line boundary of the sets closely follows the true curve swept by the vehicle and since these sets are usually only used when manoeuvring in tight situations and therefore for a short distance the advance distance for these sets has been slightly reduced to 0.6m. The negative big set, covering the tightest turns, has a 0.3m long total inhibit set with a further 0.3m using an inhibition value of 0.001. This very low but non-zero inhibition value allows the vehicle to steer close to obstacles if essential to reach a clear path, such as tight corners. These common total and near total inhibition sets are shown in Figure 6.13.

The avoidance set generation program therefore allows a wide variety of sets to be produced and their performance compared. The next section discusses the tests which were performed using various definitions and examines how different factors such as the width, length and inhibition value affect the overall avoidance performance.

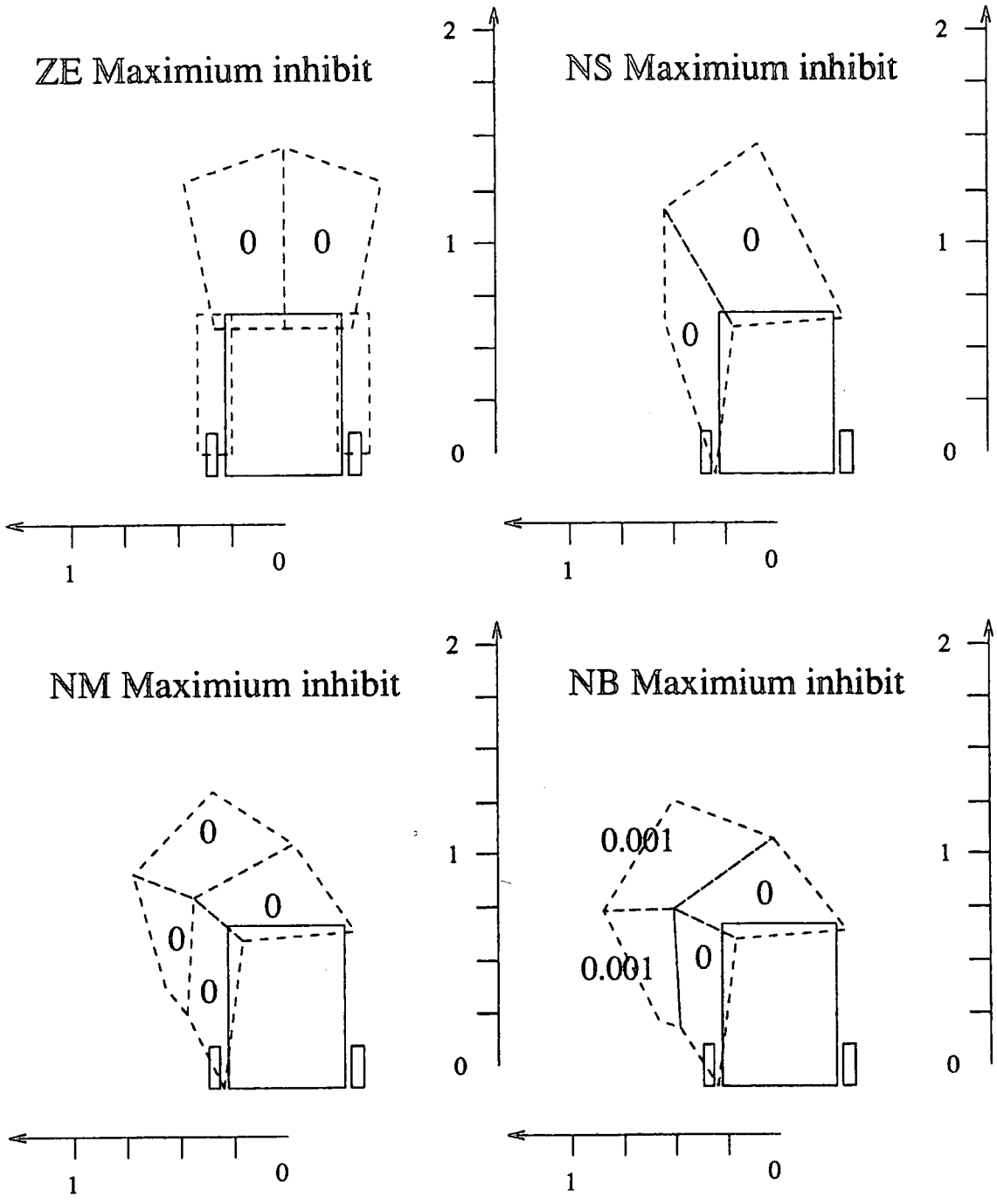

Figure 6.13: Common Total Inhibition Avoidance Set Areas.

6.7 Comparison of Obstacle Avoidance Sets

The effectiveness of a particular set of avoidance sets is linked in with all the other features of the controller making it very difficult to predict the behaviour of a particular set of definitions. In addition to this, different layouts of sets may give good results for certain classes of obstacles but bad results for others. It is almost impossible, therefore, to claim that a particular collection of avoidance sets is optimal and it would be premature to do this before tests have been undertaken using sensor acquired data to be able to include the effect of uncertainty in the conclusions. What has been undertaken therefore is an evaluation of a group of over forty potential definitions to discover some obstacle avoidance sets which provide a reasonable performance in a wide variety of situations. This section discusses how different approaches to avoidance sets affect the vehicle performance using simulation results to justify the conclusions. Finally, results obtained from the test vehicle using the most effective definitions are presented.

The tests which were carried out made use of dynamic windowing and variable rule spreading with k_{max} set to 0.7. The tests involved the vehicle negotiating a very small obstacle, 0.2m wide, a medium sized obstacle, 1m wide, and a large obstacle, 2m wide. Further tests examined the performance of different avoidance set definitions in travelling through a 1m wide gap between two poles approaching both head on and from an angle and approaching a medium sized obstacle from an angle.

6.7.1 Set Length

The initial tests examined the effect of set lengths on vehicle performance. The definition *medium* was chosen to be of a similar length to the sets used initially, that is extending 3m beyond the total inhibition sets. The inhibition values used were 0.1 for the sets bordering the total inhibition sets, then 0.3 and finally 0.5 for the most distant sets. The total distance the sets extended from the total inhibition sets was reduced for the turning steering angle sets by 10% for the small steering angles, 20% for the medium steering angles and 30% for the big steering angles to give earlier warning of obstacles in front of the vehicle than those far to one side. The resulting shapes of avoidance sets compared with the original definitions can be seen in Figure 6.14. Two other definitions *long* and *short* were used which were scaled from the *medium* definition to extend 4m and 2m beyond the total inhibit sets

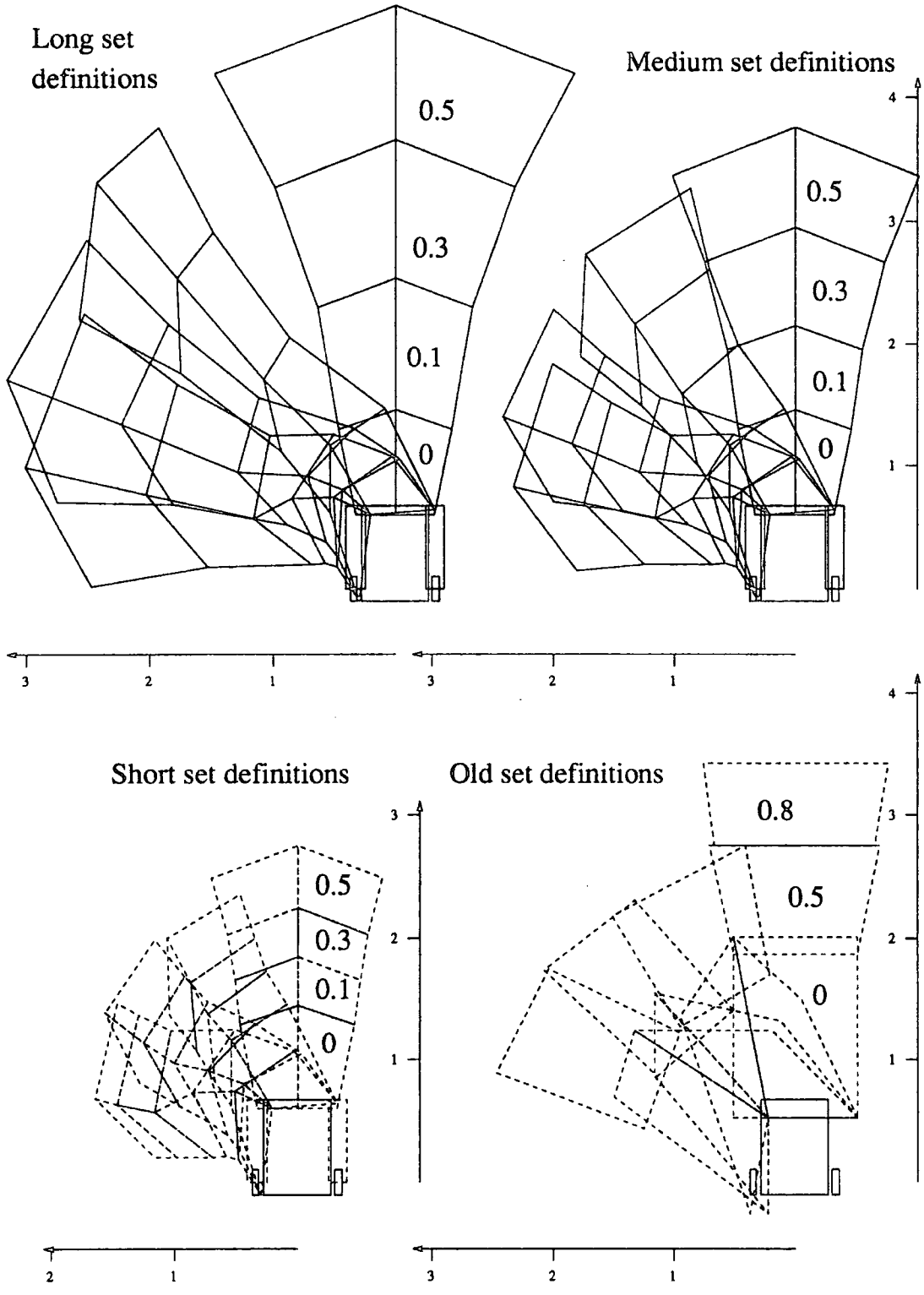

Figure 6.14: Initial Definition and Definitions of Varying Length.

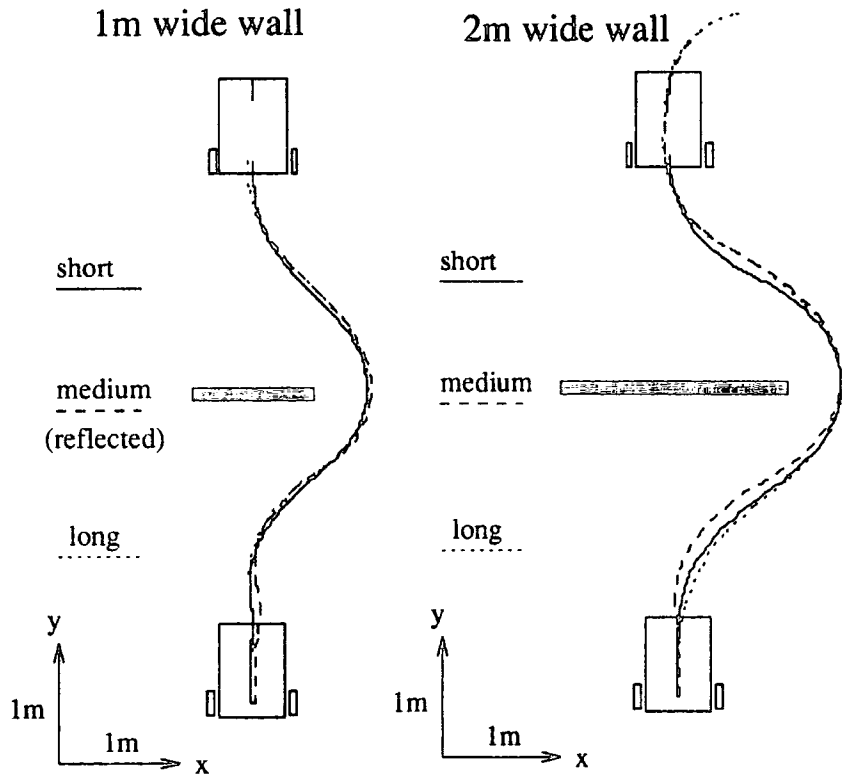

Figure 6.15: The Effect of Total Set Length on Avoidance Performance.

respectively.

Figure 6.15 shows the paths taken by the vehicle for the medium and large obstacles. Little change in the paths can be seen with the small or medium sized obstacles but with the large, 2m wide, wall only the *short* definition successfully reached the goal position and orientation on the first attempt. The main reason for the failure of the longer set definitions to complete the path is because they do not turn sharply enough back towards the goal once they have negotiated the wall. This is caused by the inhibition of the hard left steering angle sets by the wall which is then to the left of the vehicle. Clearly looking too far from the vehicle itself causes problems as possible steering angles can be unnecessarily inhibited. It was for this reason that the *short* definition was used as the basis for further tests examining the width and inhibit values of the sets as well as being used for the development of the spreading and windowing strategies described in Sections 6.3 and 6.5.

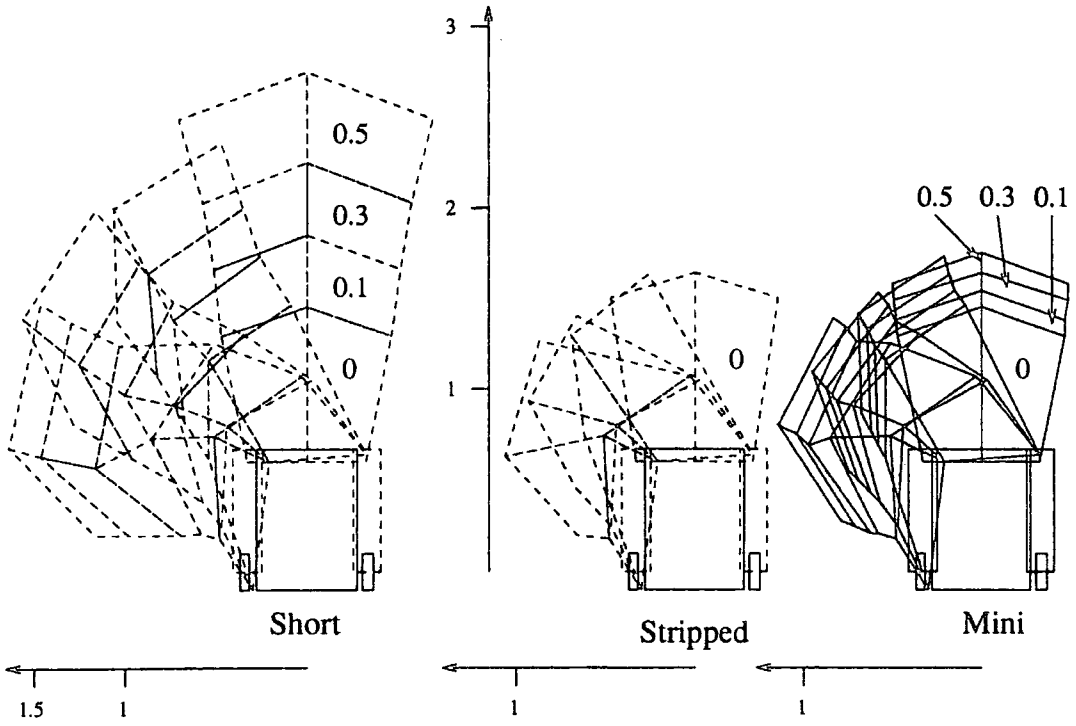

Figure 6.16: Minimal Avoidance Set Definitions.

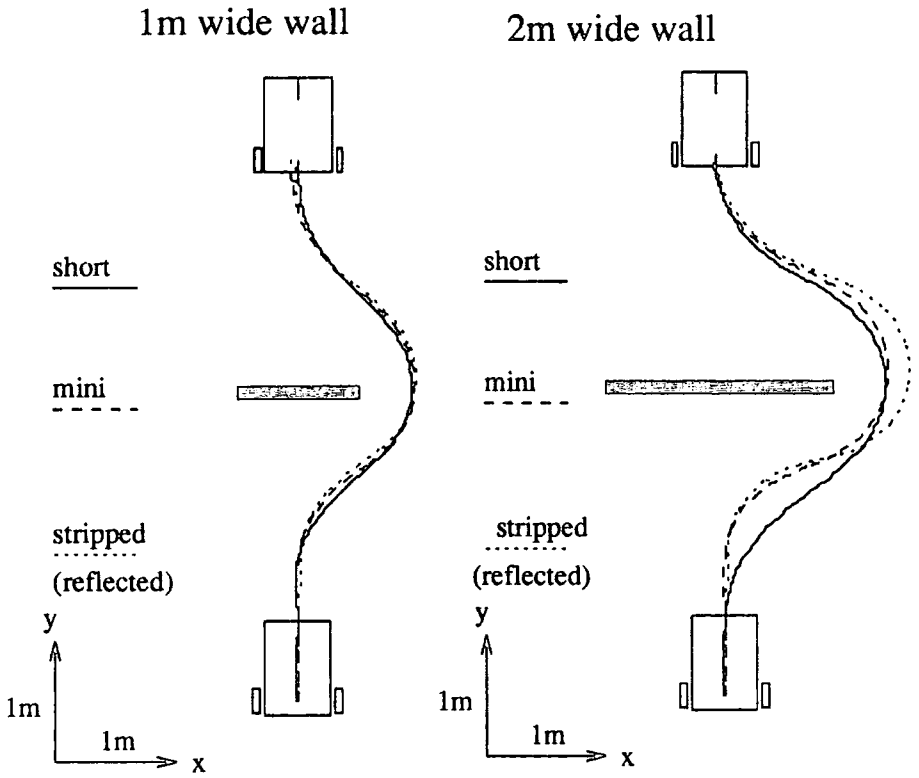

Figure 6.17: The Performance of Minimal Sets.

6.7.2 Minimal Sets

Before discussing the results of the tests based around the *short* definitions it is worth examining the performance of the vehicle when the above findings are taken to their logical conclusion and very short sets are used. Results are presented for two definitions, firstly *mini* which has the same inhibition values of 0.1, 0.3 and 0.5 as in the above cases but only in 10cm bands extending from the total inhibition sets and secondly a stripped down set which consists of only the total inhibit sets extended by 20cm. This extension was discovered to be necessary to prevent collisions when the earlier warning given by partial inhibition sets was removed. The size of the resulting sets compared with the *short* definition can be seen in Figure 6.16.

Figure 6.17 shows the path taken by the simulated vehicle around 1m and 2m wide walls using these minimal definitions compared with the *short* definition used in the previous tests. The difference round the 1m wide wall is not very marked, the stripped definitions steering away slightly later and consequently back a little later. The main difference is that the stripped definitions while taking a slightly longer path reach the goal in slightly less time. The main reason for this seems to be the lack of uncertainty in their paths. Although the longer definition steers out earlier there is a little oscillation as it approaches the obstacle and it is this which causes the vehicle to slow. When considering the larger obstacle however there is little doubt that the minimal sets suffer from short-sightedness. They approach too close to the wall and have to turn sharply to avoid it rather that taking the faster, wider sweep of the longer set definitions. The advantage of having higher inhibition values at the extremes of the sets are shown by the tighter curve around the edge of the obstacle taken by the *mini* definition which occurs because, unlike the fully stripped definition, it is not hampered by the extra length of the hard left total inhibition set.

An interesting conclusion which can be drawn from these results is that the partial inhibition sets can be shrunk or even removed without causing a catastrophic failure. It may be that if computation time was at a premium, or data far from the vehicle unreliable, these simple definitions might be the best. This would particularly be the case if the larger obstacles were detected by a higher level in the control hierarchy and assistance given to the navigation controller in negotiating them, perhaps by the use of sub-goals as discussed in Section 7.4.

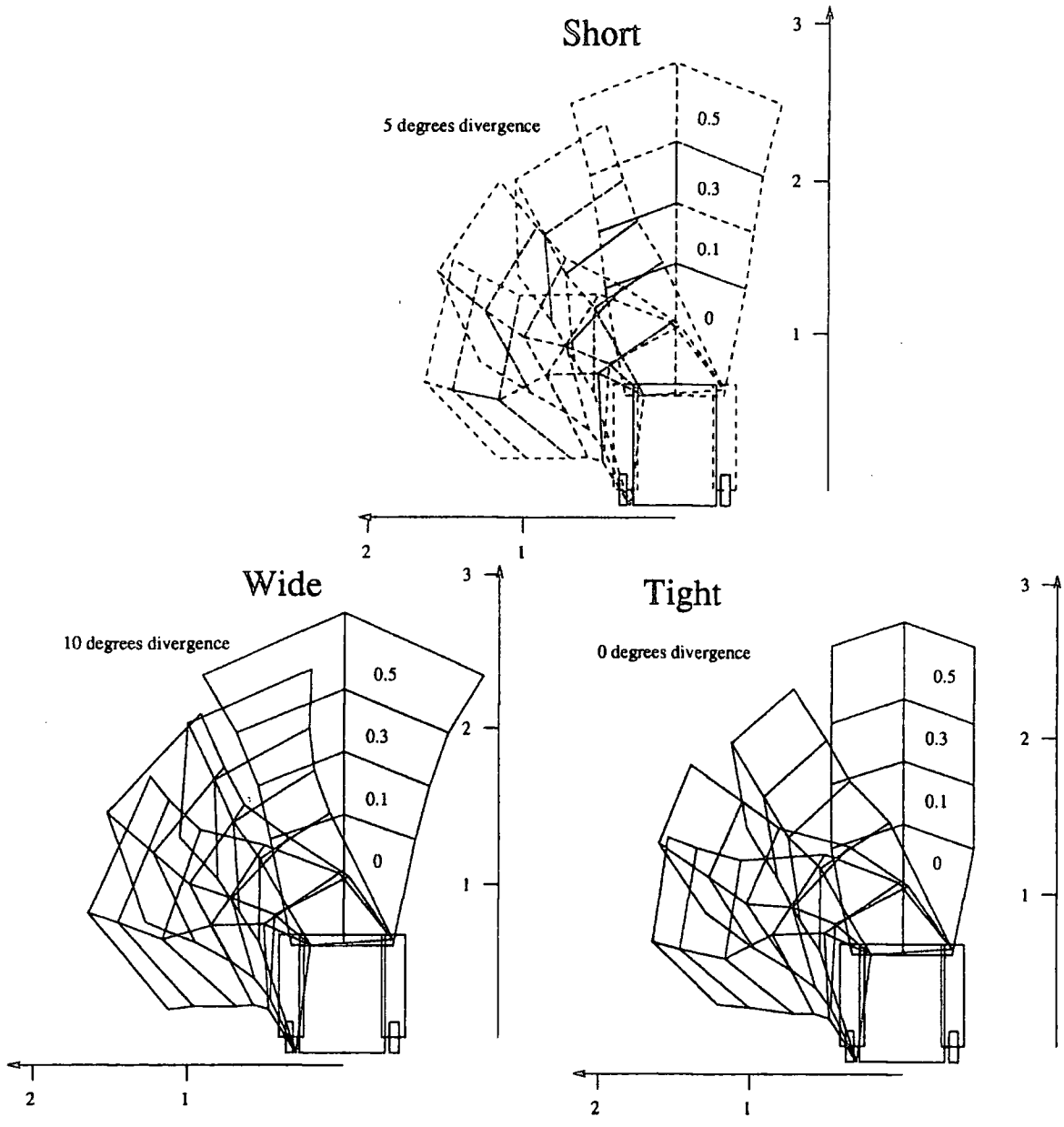

Figure 6.18: Avoidance Set Definitions of Varying Widths.

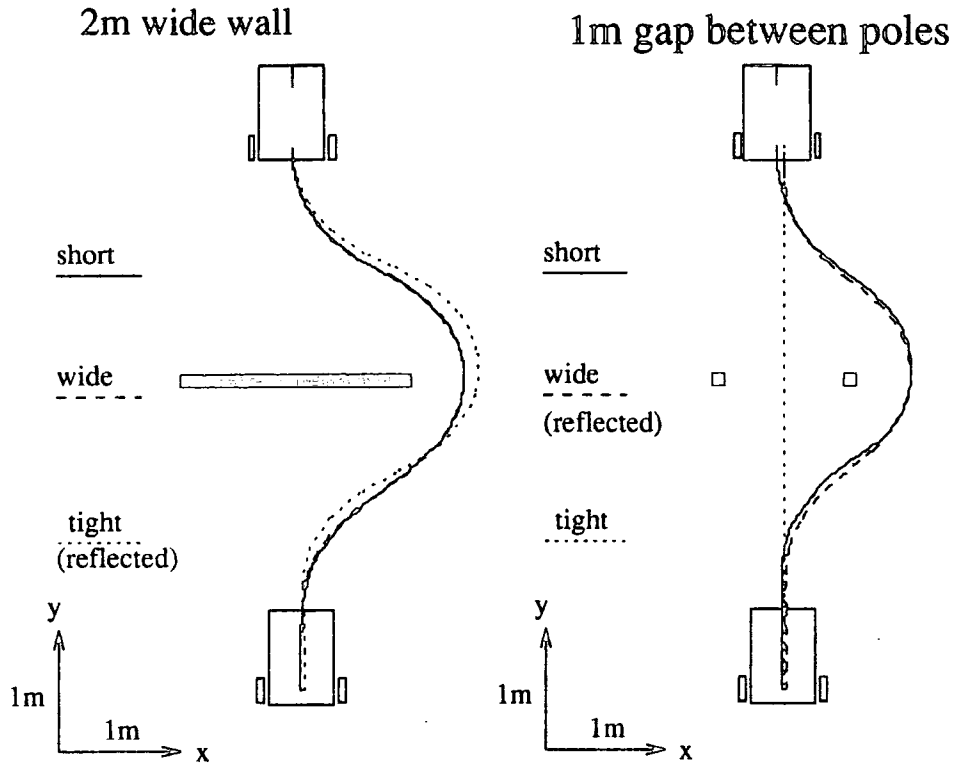

Figure 6.19: The Effect of Varying Set Widths.

6.7.3 Set Width

With the length of the *short* definition shown to give a good response both for small and large obstacles the next important consideration is the width of the sets. As has been explained previously the avoidance set generation program allows for the sets to expand with distance by allowing the extremes of the sweep lines to follow a path angled differently from the centroid value of the steering angle set with which they are associated. The widening of the sets represents the uncertainty as to the precise angle which the vehicle will steer on since it is always a defuzzified combination of several sets and if an obstacle lies just at the edge of the area which would be occupied by a vehicle steering at the centroid angle it should be considered as a potential source of a collision. The problem however is that by widening the sets the ability of the vehicle to pass close to obstacles and to detect narrow gaps is reduced.

The *short* definition includes a 5° spreading of the sweep line edges and to examine the effect of set width two other definitions were used *tight* and *wide* with 0° and 10° of divergence respectively. The shapes of these avoidance set definitions relative to one another can be seen in Figure 6.18. The trade off between tight and wide sets can be seen

in Figure 6.19 where it can be seen firstly that the *tight* definition causes the decision to steer around a large obstacle to be made later than with the wider definitions because the reduced width encourages the vehicle to travel closer to the obstacle, in this case a bad decision. In the case where the vehicle has to travel through a small (1m wide) gap between two poles the *tight* definitions allow the vehicle to ignore the proximity of the poles and travel through the gap between them. Both the *short* and *wide* definitions take a path around the poles since the spreading of the sets causes the gap between the poles to be considered as a potential obstacle. Even though the total inhibition sets are narrower than the gap the decision to steer around it is made earlier. It is also noticeable that the *wide* set definition causes the vehicle to reject the possibility of the gap earlier than the *short* definitions. Another problem which was noted in tests around smaller walls was an increase in oscillations caused by dynamic indecision when using the tighter set definitions, caused by the gaps introduced in the coverage and reduced overlap of the sets. From these results it would seem that the divergence angle of 5° is a reasonable compromise but it would be more desirable to have set definitions capable of both negotiating small gaps and steering out early to go around large obstacles.

6.7.4 Inhibition Values

Another important element in the avoidance scheme is the selection of inhibition values. The non-zero inhibition values represent how seriously a distant object is to be treated with respect to the likelihood of the vehicle colliding with it. Without a time consuming analysis of the future path of the vehicle it is difficult to assign a value to this and as the vehicle moves, an object may move or turn out to be caused by a spurious sensor reading. The previous tests were carried out with the inhibition values 0.1, 0.3 and 0.5. These give a spread from 0.5 which in practice only has a small effect on the final output to 0.1 which all but guarantees that the associated steering angle will not be selected. To examine the effect of altering these values several tests were carried out using more optimistic or pessimistic values by multiplying the inhibition values by a constant between 0.5 for the most pessimistic and 1.5 for the most optimistic. Tests were also carried out to look at the effect of varying the inhibition values with respect to steering angle, having pessimistic sets in front of the vehicle, becoming more optimistic for the harder steering angle sets and vice versa but this did not produce any noticeable benefits.

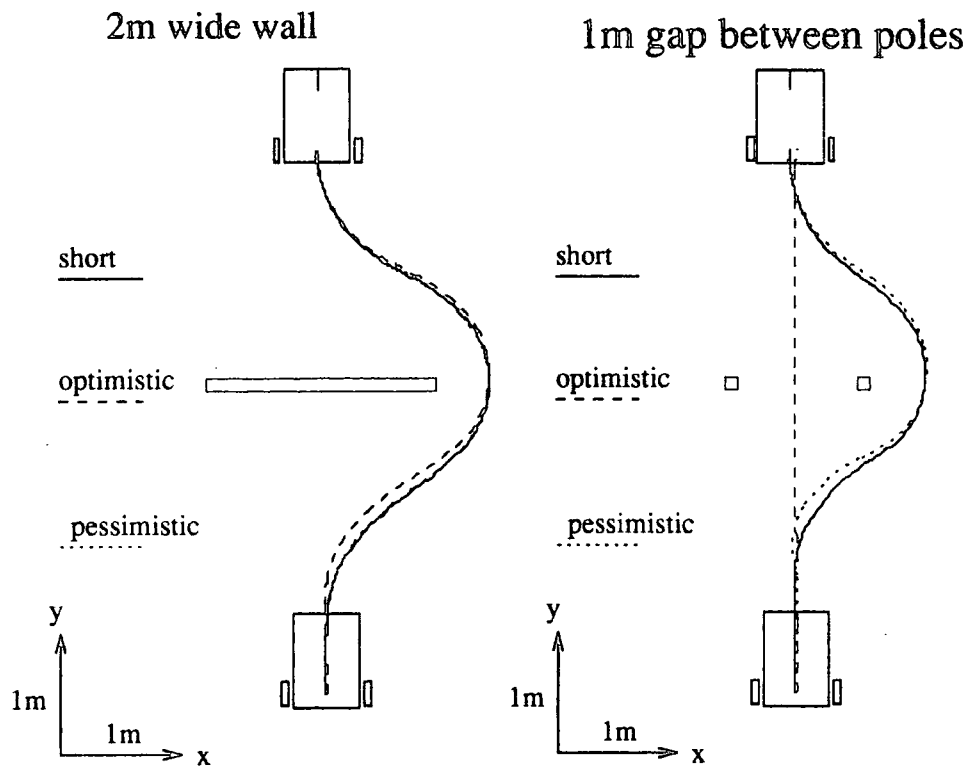

Figure 6.20: The Effect of Varying the Inhibition Values.

Figure 6.20 shows how the performance was changed with inhibition values reduced by 25% in the pessimistic case and increased by 25% in the optimistic case. The higher the numerical values in the mask vector the less of an inhibitive effect they have when combined with the navigation fuzzy fit vector. The test round a large obstacle shows that the pessimistic case is almost identical to the normal (*short*) one while the optimistic case steers out a little later. The more optimistic path has less early oscillation which gives an increase in speed. This is because the weaker inhibition values give rise to smoother changes which enable the path to be taken at higher speeds. The major improvement which the more optimistic inhibition values produce is the better performance through the narrow gap between poles. As can be seen from the figure the more optimistic values enable the vehicle to adopt the shorter route through the gap. Surprisingly the more pessimistic definition steers away from the gap later than the *short* definition. This is due to indecision about which way to turn, caused mainly by the large changes in inhibition values inducing oscillation.

With even more pessimistic sets this oscillatory problem became worse, further slowing the vehicle. With more optimistic values than the ones shown here the path round the large obstacle was severely degraded with the turn away from the obstacle being left very

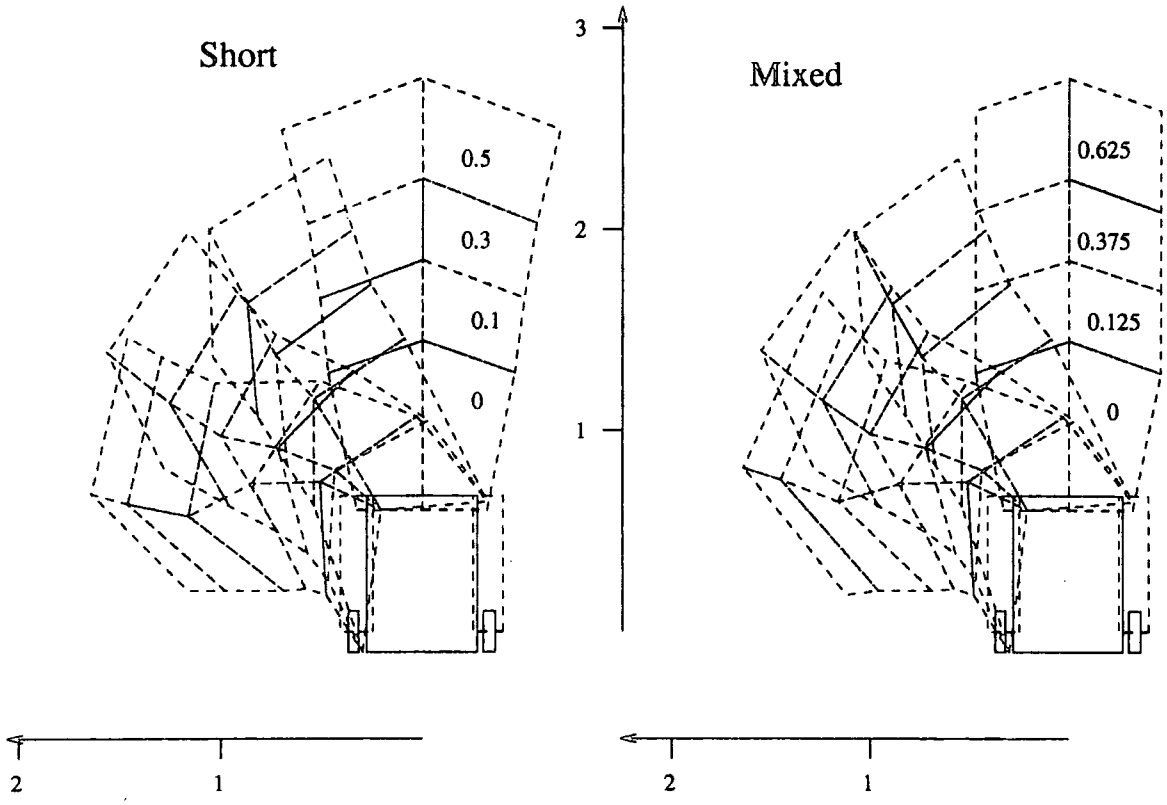

Figure 6.21: The Final Avoidance Set Definitions.

late, resulting in an unsatisfactory trajectory. Over several other tests the slightly more optimistic avoidance set values proved to give faster, smoother runs than the previous settings and to be better able to cope with gaps. Avoidance sets with inhibition values 10% and 20% more optimistic than the *short* definitions were seen to give the smoother performance but not be sufficient to allow the vehicle to negotiate the gap so the inhibition values of 0.125, 0.375 and 0.625 were adopted as giving the best compromise in performance between large obstacles and small gaps.

6.7.5 The Final Avoidance Set Definition

From the previously described work it can be seen that the definition of avoidance sets is a compromise between being able to take early avoiding action in the case of large obstacles and not being turned away from obstacles so early that potential gaps are ignored. The tests have shown that set definitions extending for two metres in front of the vehicle give reasonable performance for large obstacles although very short sets extending only 1m give slightly better results for smaller obstacles. Tests have also shown that in order to be able

to negotiate small gaps the set definitions need to be spread very little or have less of an inhibiting effect. The divergence of the set boundaries gave the best results with a setting of 5° although less divergence improved the performance through narrow gaps. A compromise between the need to spread out the coverage of sets to avoid turning too close to obstacles and the need for the ability to travel through tight gaps gave rise to a compromise definition which has divergence angles (that is the difference in steering angle used to trace the path of the extreme edge of a set compared to the centre) of 0° for the straight steering sets (ZE) rising through 3° and 6° to 10° for the larger steering angle sets. This definition can be seen in Figure 6.21 and gave slightly improved performance over the best optimistic version of the *short* definition. This definition labelled *mixed* has been adopted for use on the vehicle as it gave the best performance over a series of tests, bearing in mind the path followed, time taken and smoothness of control. The next section presents results from these tests comparing the actual performance of the research vehicle with the results of the simulated tests.

6.8 Obstacle Avoidance Results

This section presents the results of tests carried out using the research vehicle rather than the simulation results used for the development work described in this chapter. The tests were all carried out using dynamic windowing with a maximum window width of four and variable rule spreading with k_{max} set to 0.7. The anti-oscillation features described earlier in the chapter are operating and the *mixed* obstacle avoidance set definitions were used.

Figures 6.22 and 6.23 show the vehicle avoiding the one and two metre wide walls which have been commonly used as examples. Figures 6.24 and 6.25 show the vehicle coping with slightly different situations, firstly avoiding a 1m wide wall when approaching on a curve rather than a straight line and secondly when the wall is asymmetrical with respect to the normal path the vehicle would take. Finally the ability of the vehicle to negotiate a narrow gap is shown in Figure 6.26. In all these cases it can be seen that the vehicle follows a path similar to but not exactly the same as the simulated path due to the uncertainty associated with the location system of the vehicle. All these figures show the line followed by the reference point on the vehicle, the centre of the rear wheelbase, in addition to a series of vehicle positions which gives an idea of the clearance between the vehicle and an obstacle.

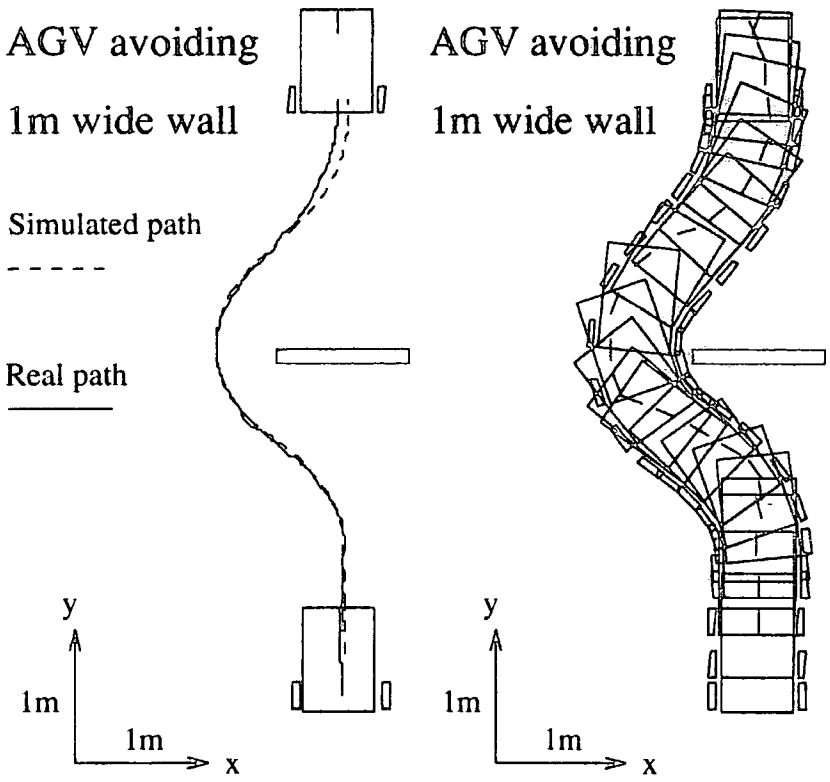

Figure 6.22: Results for Avoidance of a 1m Wide Wall.

All these results combine to show the success of the control method in avoiding a variety of obstacles. The next chapter considers problem situations which the controller cannot cope with and the developments which might be applied to enable it to do so.

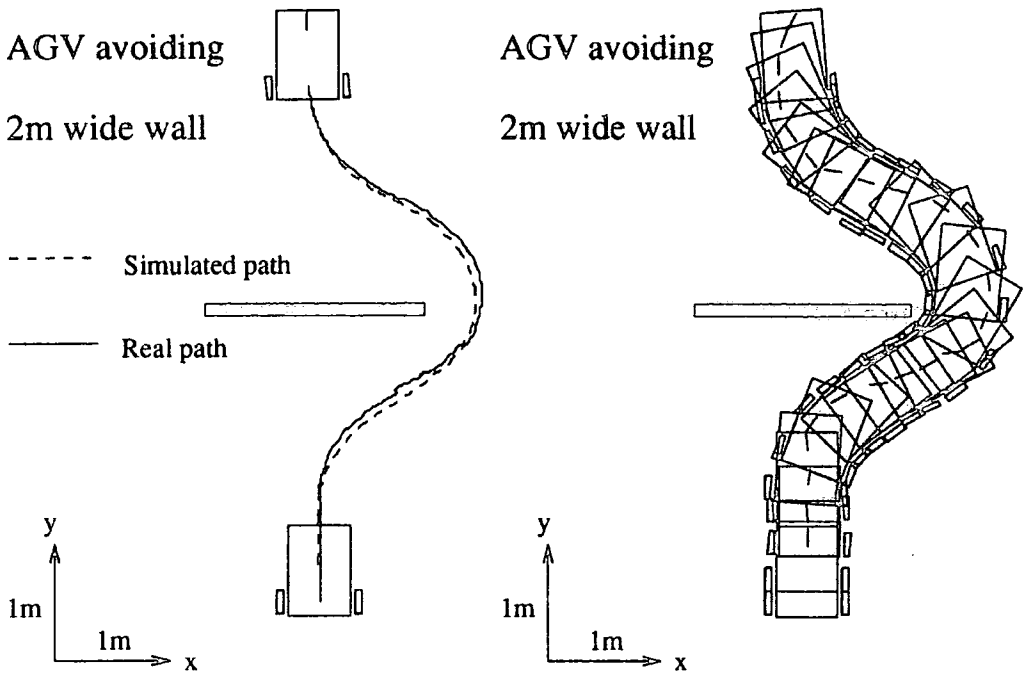

Figure 6.23: Results for Avoidance of a 2m Wide Wall.

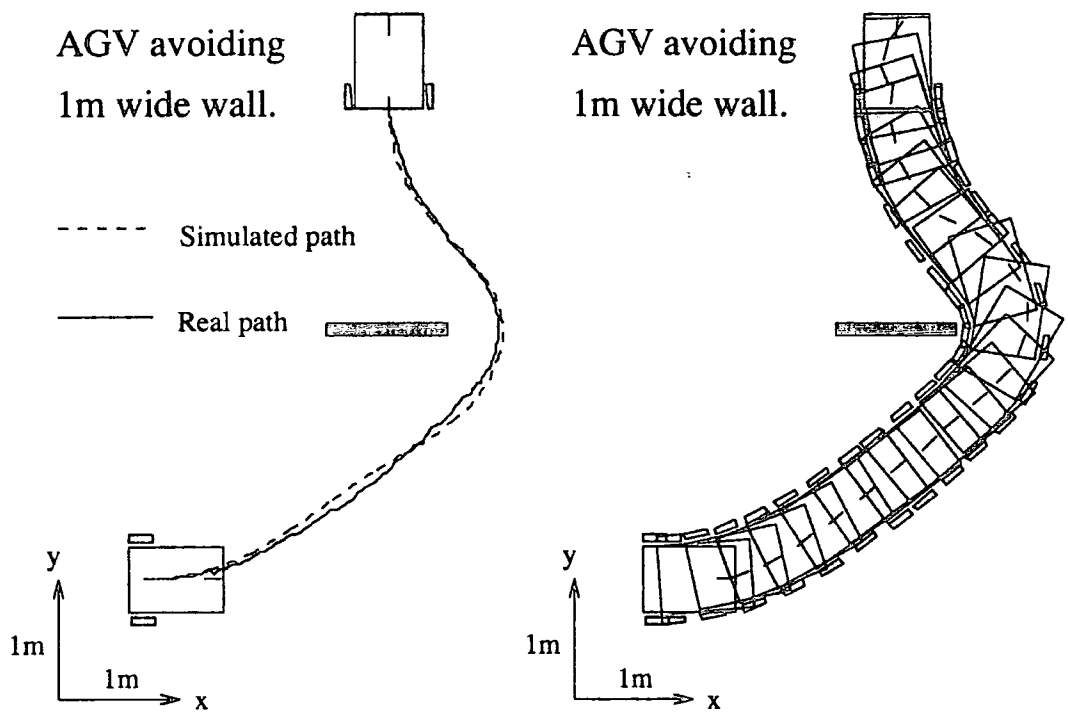

Figure 6.24: Results for Avoidance of a 1m Wide Wall Approaching From the Side.

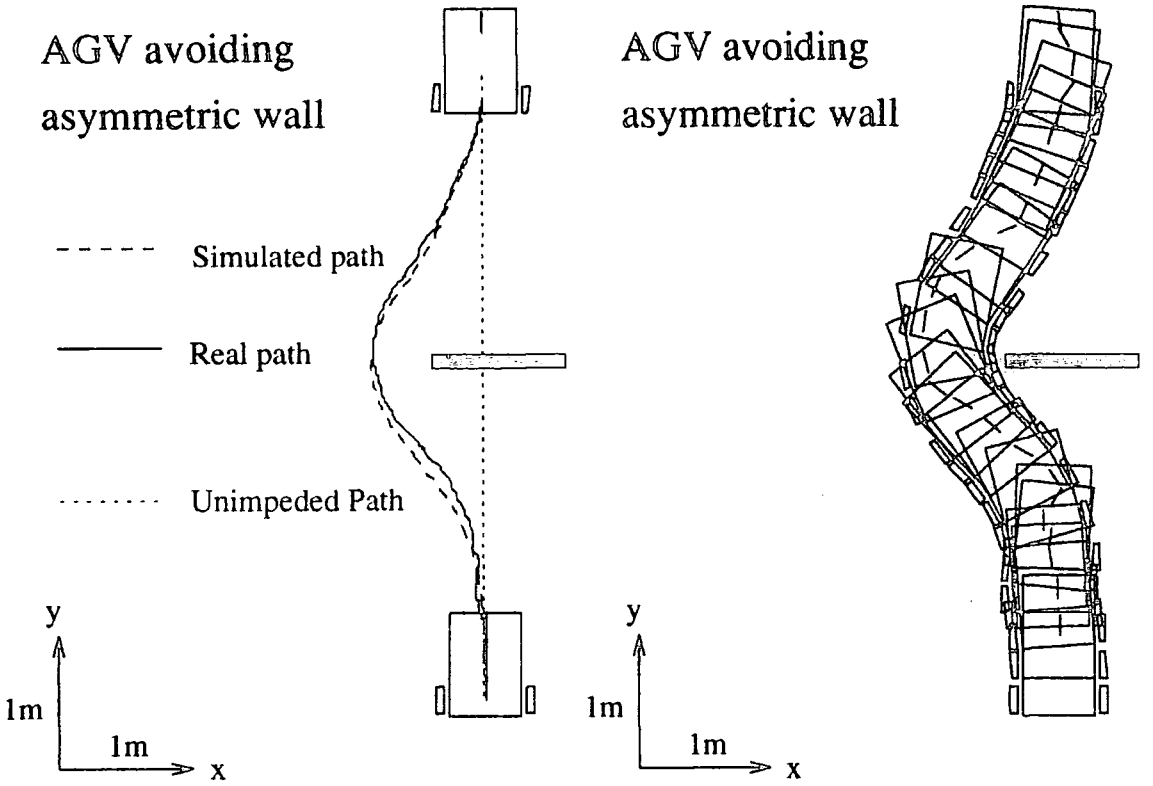

Figure 6.25: Results for Avoidance of an Asymmetric Wall.

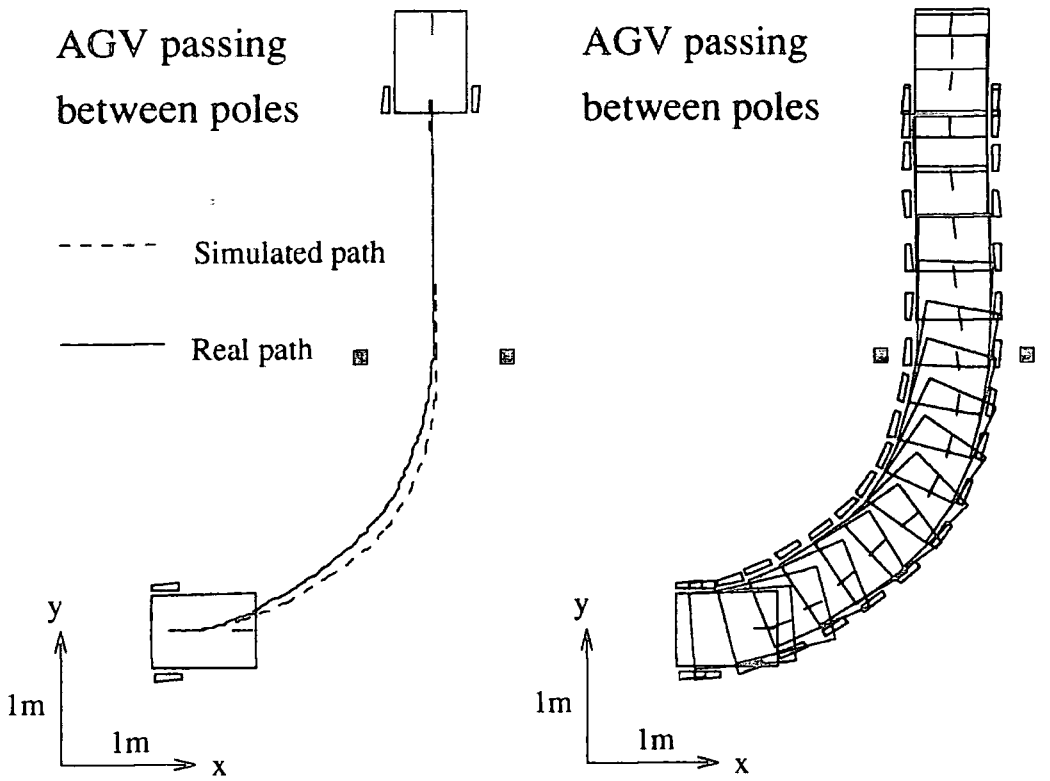

Figure 6.26: Results for a Narrow Gap Between Poles.

Chapter 7

Performance of the Final Controller

The results given at the end of the previous chapter show that the vehicle is capable of navigating around single obstacles on the way to its goal. Since the controller was designed to navigate the vehicle through a clear area it would be unreasonable to expect it to be capable of doing much more than avoiding a few intermediate obstacles. Section 7.1 shows the controller dealing with a complex arrangement of many obstacles. This demonstrates the strength and versatility of the technique in that some more complex manoeuvres can be successfully completed without the intervention of a sophisticated planning stage. Following this Section 7.2 discusses how the sampling frequency at which the controller operates affects its present performance and briefly comments on how using sensor acquired data may affect this.

Section 7.3 introduces some problem situations where the basic controller is no longer appropriate and discusses how a higher level planning stage might try to alleviate these problems. The way in which sub-goals provided from a higher level of the control structure could be used to improve performance and cope with situations which are beyond the power of the basic method is illustrated in Section 7.4. Finally, some tests undertaken within the laboratory using sub-goals and an occupancy grid including all the fixtures in the laboratory in addition to discrete objects are presented. It is these tests which have been recorded on video to provide a visual record of the vehicle in action.

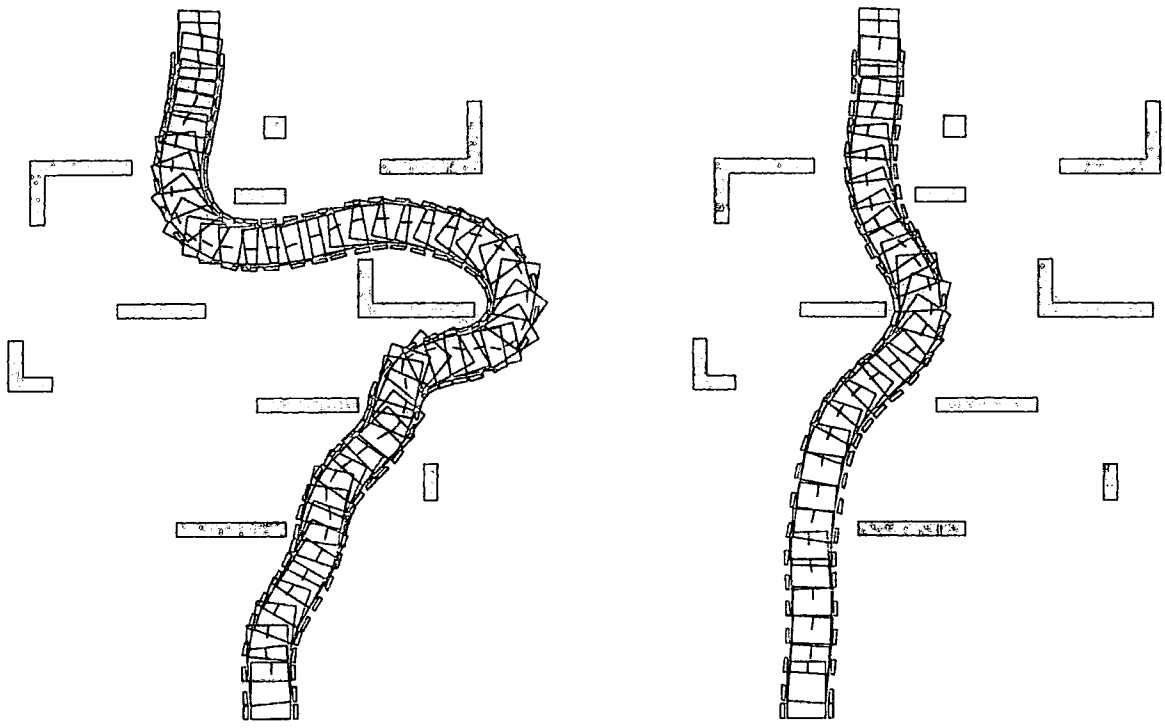

Figure 7.1: Vehicle Navigating Through a Jumble of Objects.

7.1 Avoidance of Multiple Objects

The avoidance method is capable of dealing with a complex jumble of obstacles and travelling through them to the goal. To present an example of such a case the path produced by the simulator navigating through a jumble of objects to the same goal from a series of different starting positions is displayed in Figures 7.1 and 7.2. In all four of the examples the arrangement of obstacles and the goal position and orientation is the same but the starting position is different. The four examples show how the controller manages to drive the vehicle on a number of different paths all of which end up satisfactorily at the goal.

Another more complex situation is shown in Figure 7.3 which shows the vehicle successfully negotiating a tight corner in a corridor to reach its final goal. This is a difficult situation since the clearance is very small. The selected obstacle avoidance definitions manage the turn which stumped many of the other possible definitions. The figure shows the vehicle negotiating the corner when the goal is at the far end of the corridor and when it involves making another 90° turn at the end of the corridor.

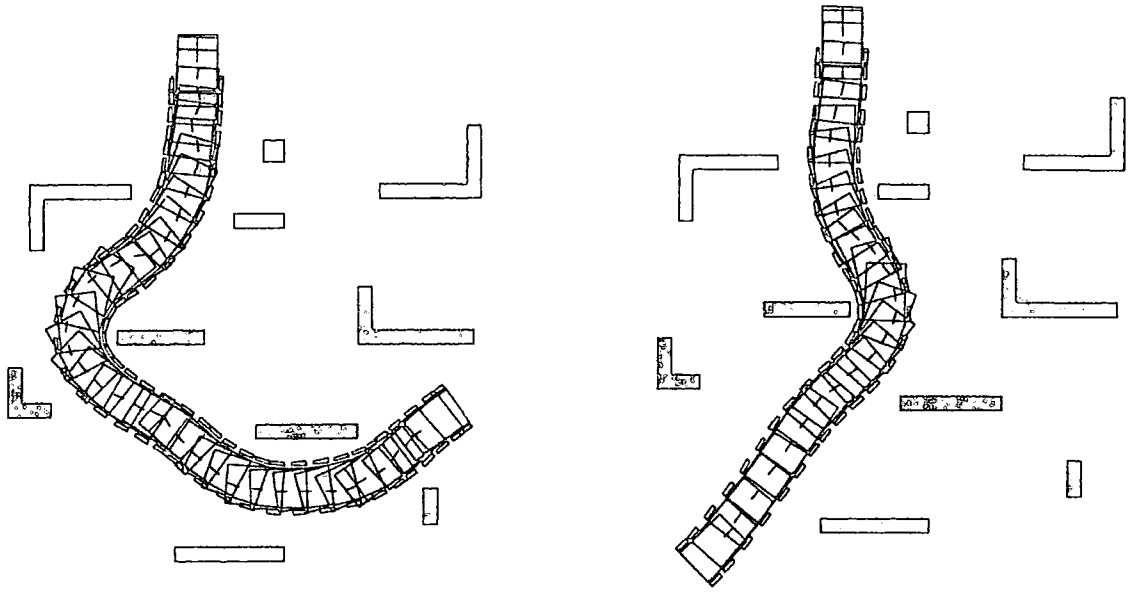

Figure 7.2: More Paths Through a Jumble of Objects.

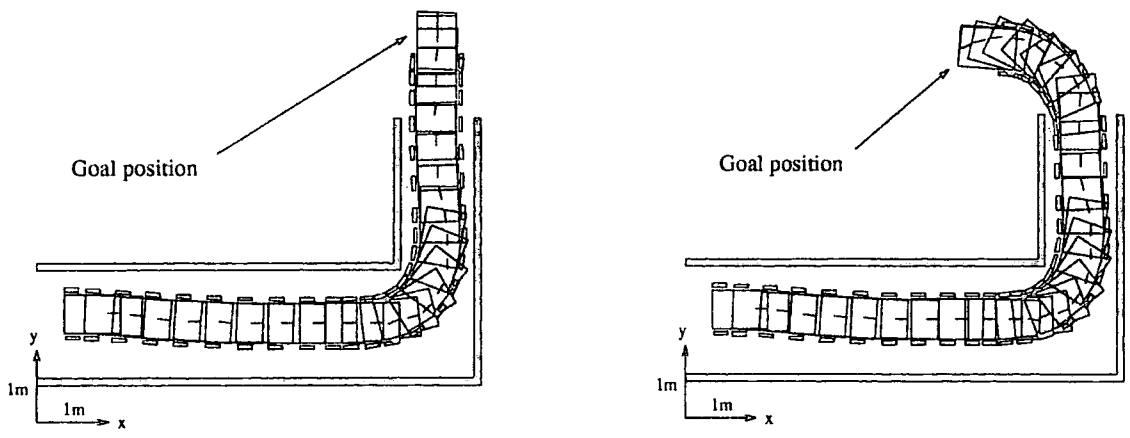

Figure 7.3: AGV Negotiating a Tight Corner in a Corridor.

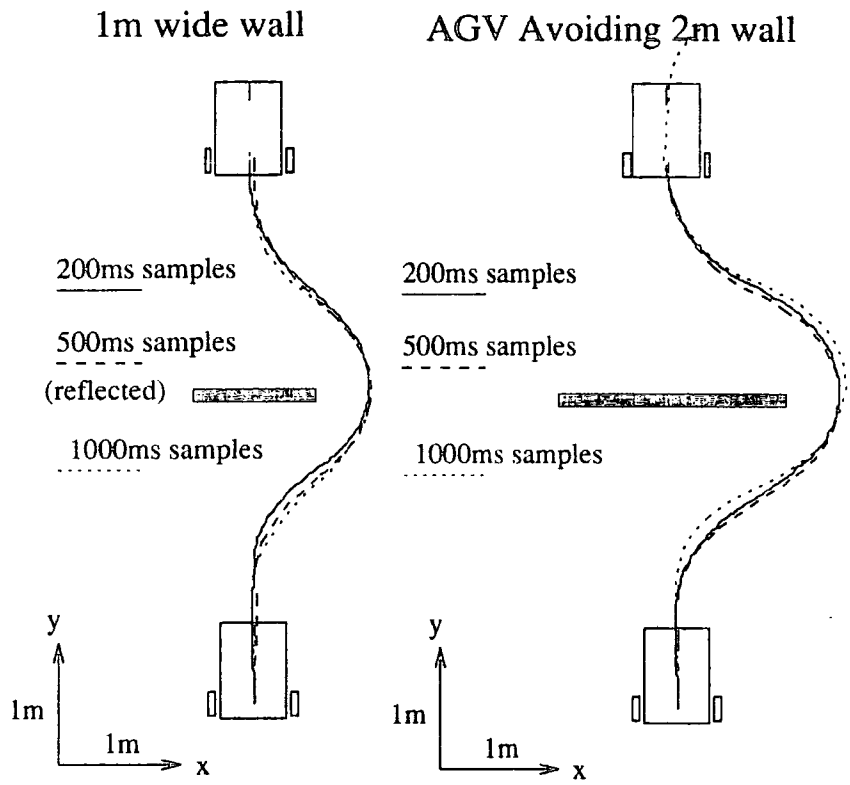

Figure 7.4: The Effect of Sample Time on Avoidance Performance.

7.2 Controller Sampling Frequency

During the tests described in this thesis the main control loop activated every 200ms. Early tests had shown that the controller was not greatly affected by changes in sample times and this value was used since it corresponded to a small, 2cm, step during the initial fixed velocity tests. This meant that the vehicle should be able to arrive within 2cm of the goal, while increasing the sample time would increase the error at the goal position. As far as the performance of the controller during a run is concerned the sample time has little effect on the path as can be seen in Figure 7.4.

One of the problems which this figure shows is that with long sample times it is possible to pass the goal without realising it, as has occurred in the case of the 1000ms sample time example avoiding the 2m wide wall. Another problem occurs in situations where the clearances are very tight as in the corridor shown in Figure 7.3. Repetitions of this test with longer sample times showed that they cause the vehicle to fail to navigate the corner because of the delay in starting the turn.

The advantage of longer sample times is that they reduce the computational load on the

computer system without drastically affecting the controller performance. The problems, however, are that it makes it harder to achieve a precise goal position and more likely to fail in very difficult situations with tight clearances. An additional problem is that new information may be perceived by sensors but this cannot be acted upon until the next sample interrupt. Therefore while modifications to the final docking stage and the use of sub-goals might alleviate the first two problems without the inclusion of sensor data rates it is difficult to say whether or not it is justified to increase the time between samples. For this reason the sample time was kept at the fairly short time of 200ms.

7.3 Problem Situations

There are a number of problem situations which can prevent the vehicle from reaching its goal. Firstly there are of course situations where the goal is so surrounded by objects that it is physically impossible to reach. This is clearly the job of a mission planner to recognise and assign a new goal (or move the objects!). The second class of problems are those which do have a solution but the controller fails to find this solution. There are two ways in which the controller can fail in these situations. It can become locked on a path which never reaches the goal but keeps following a closed loop or it can manoeuvre the vehicle into a position so that it cannot move forward without a collision (for a discussion of the issue of reversing see Section 8.3).

7.3.1 Wide Walls

An example of the closed path problem can be seen in Figure 7.5 which shows the vehicle following a figure of eight when faced with a wide wall. The problem is not that the wall blocks all possible paths to the goal, it is simply that the controller does not manoeuvre the vehicle into a position from which it could negotiate the obstacle. The problem is that the wall is so wide that by the time the end of the wall is reached the navigation controller wishes to turn back again to line up with the goal. It has no mechanism for discovering that the path it wishes to follow is blocked by the wall even though it has previously been forced to steer away from the wall and consequently the same mistake is repeated each time.

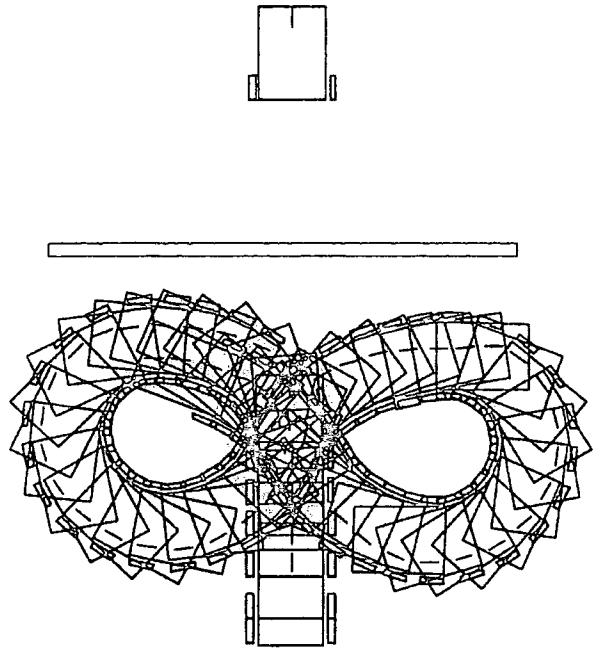

Figure 7.5: Vehicle Faced with a Wide (4m) Wall.

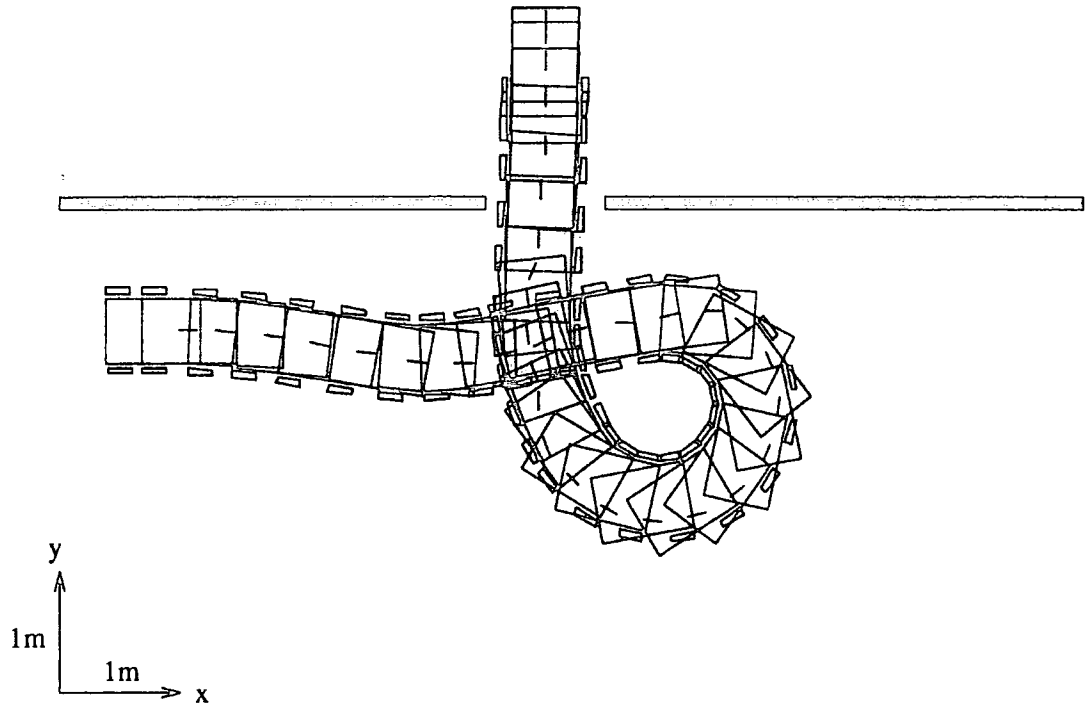

Figure 7.6: Vehicle Faced with a Narrow (1m) Gap, Goal Aligned With Gap.

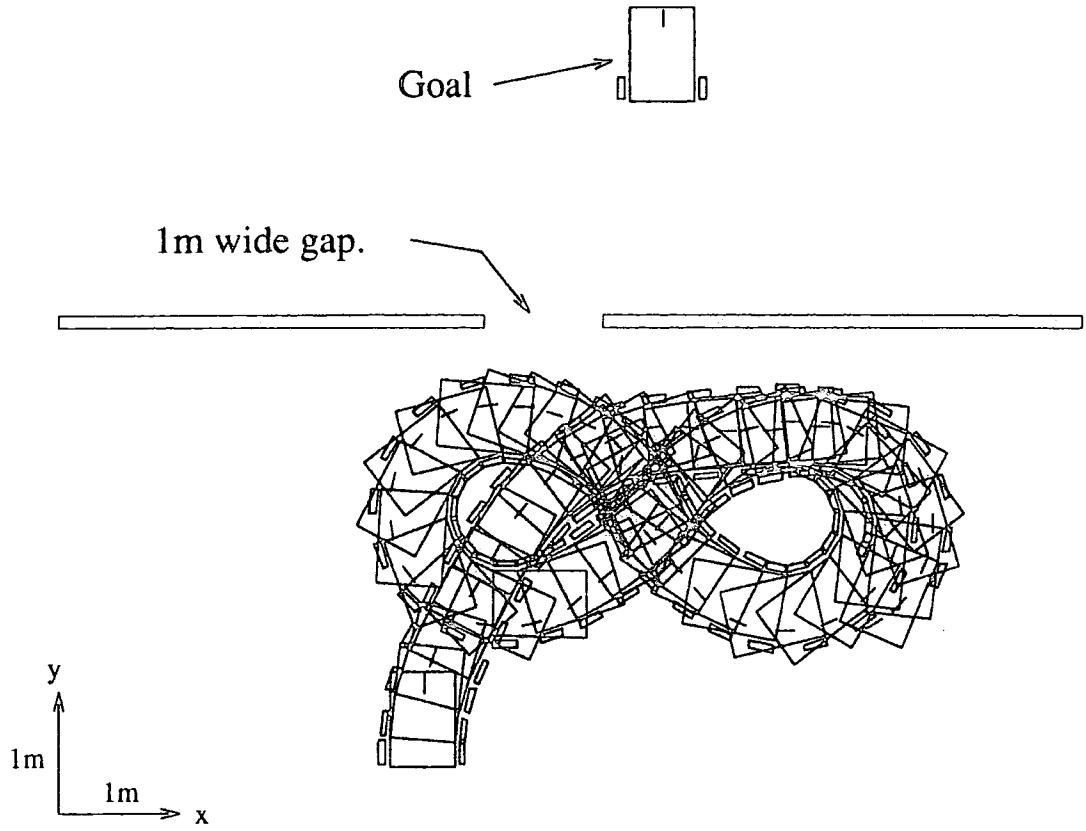

Figure 7.7: Vehicle Faced with a Narrow (1m) Gap, Goal Offset From Gap.

7.3.2 Narrow Gaps

Figure 7.6 shows the vehicle having to make a loop as it approaches a narrow gap. While this is not a complete failure because the vehicle does reach the goal it does make the path unnecessarily long. This is because the vehicle approaches at too sharp an angle and is unable to turn into the gap. The vehicle needs to be lined up with the gap early so it can pass through on the first attempt.

The worst effect of this inability to predict the required approach to a narrow gap can be seen in Figure 7.7 where the goal is slightly offset from the gap. The vehicle approaching from below lines up with the goal not the gap and turns away from the wall too late to be able to turn sideways into the gap. The result is a figure of eight loop similar to the one produced by the very wide wall.

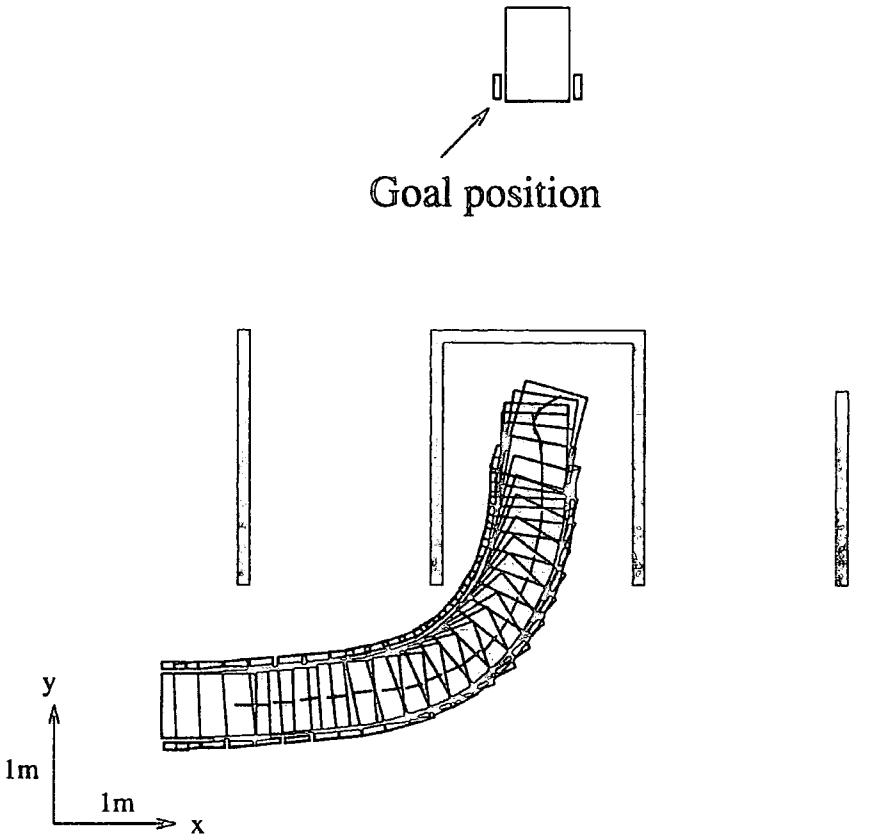

Figure 7.8: Vehicle Reaching a Dead End.

7.3.3 Dead Ends

Since the vehicle considers only a limited amount of information about obstacles it can take a path which while it avoids obstacles in the short term guides the vehicle into a position from which it cannot escape. Figure 7.8 shows such a situation. The vehicle has two possible routes to the goal up corridors but the third, central, corridor is blocked at the far end. The controller only sees that the path directly to the goal is clear for the space immediately in front of the vehicle and so guides the vehicle down the central corridor. At the far end the vehicle has no room to turn around and so is stuck.

7.4 A Potential Solution - Sub-goals

The problems illustrated by the previous examples have shown that the obstacle avoidance controller is just that, a system for avoiding obstacles. It cannot provide a path in all situations because its primary objective is preventing collisions, not reaching the goal. The

navigation controller is designed to guide the vehicle in an open space and does not make allowances for obstacles. What is required is an ability to recognise longer term problems and produce guidance for the controller to prevent it reaching these problem situations. It is clear that as the size and complexity of obstacles grows they should be treated as *features* which must be allowed for by a planner. The point at which an object becomes a feature rather than an obstacle is difficult to decide and is beyond the scope of this thesis. Once they have been identified, however, the controller can be adapted to cope with these situations. Saffiotti et al [47] propose the use of multiple controllers, each applicable to a different situation such as travelling down a corridor, crossing a door etc. Slack [35] proposes a different technique, the inclusion of preferred directions and turning moments into mapping information. The implementation of such techniques to guide the low level controller may form the aim of future work but a simple solution which can easily be manually implemented is the inclusion of sub-goals.

At their simplest sub-goals are the same as the overall goal but once they have been satisfied by the control system a higher level controller presents the system with a new goal. More complex sub-goals may have their influence merged, possibly using fuzzy logic, with other sub-goals and the overall goal. As Payton et al [37] successfully argue fixed sub-goals are inflexible and unable to cope with changes in the expected environment but this simplest form of higher level guidance serves to illustrate how the controller can easily cope with the problem situations presented earlier in this chapter. The following examples have therefore been prepared to illustrate the effect that sub-goals have without examining any of the wider issues surrounding their correct placement and relative importance.

The use of sub-goals with wide walls is shown in Figure 7.9 which shows a comparison between the unaided navigation around a two meter wide wall and the sub-goal assisted path and the successful avoidance of the 4 meter wide wall which caused the failure shown in Figure 7.5. In both cases it can be seen that the vehicle steers out to go around the wall very early as it tries to line up with the sub-goal.

The use of sub-goals with the two examples of narrow gaps can be seen in Figures 7.10 and 7.11 which both have a sub-goal placed centrally in the gap. The vehicle is therefore drawn to this sub-goal and manages to successfully line up with the gap to negotiate it. In the case of Figure 7.10 this means that the vehicle avoids the unnecessary loop and in Figure 7.11 it avoids the closed figure of eight loop and reaches the goal.

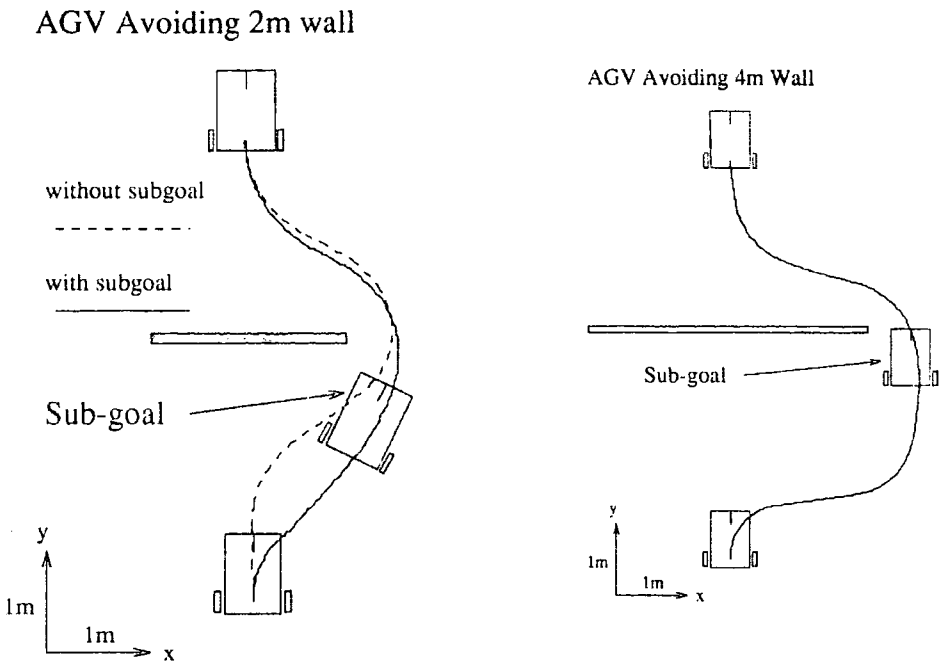

Figure 7.9: Sub-goal Navigation Around Wide Walls.

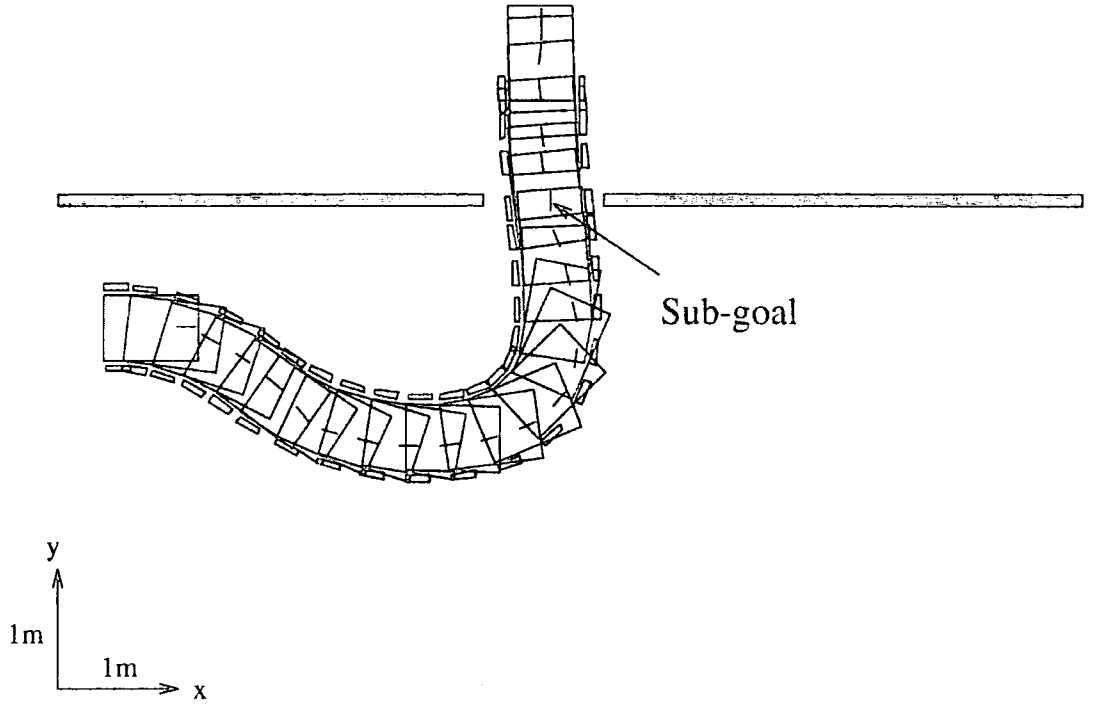

Figure 7.10: Vehicle Faced with a Narrow (1m) Gap, Goal Aligned With Gap, Sub-goal in Gap.

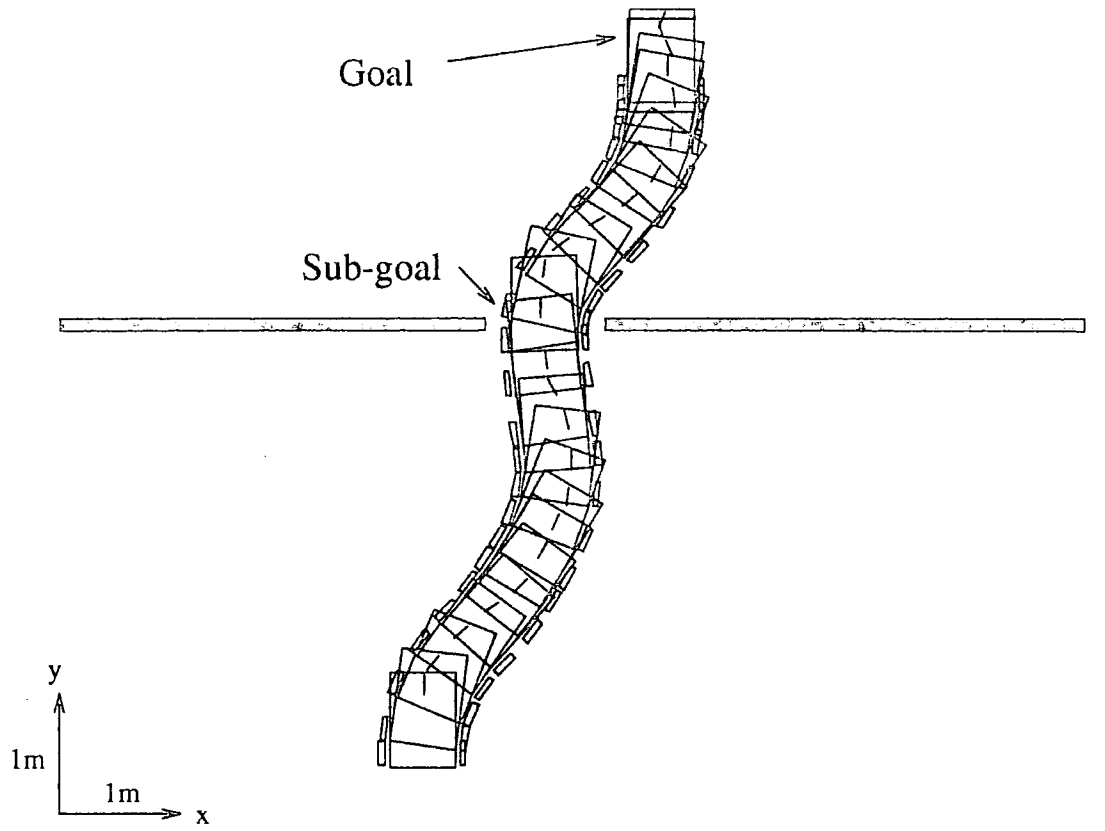

Figure 7.11: Vehicle Faced with a Narrow (1m) Gap, Goal Offset From Gap, Sub-goal in Gap.

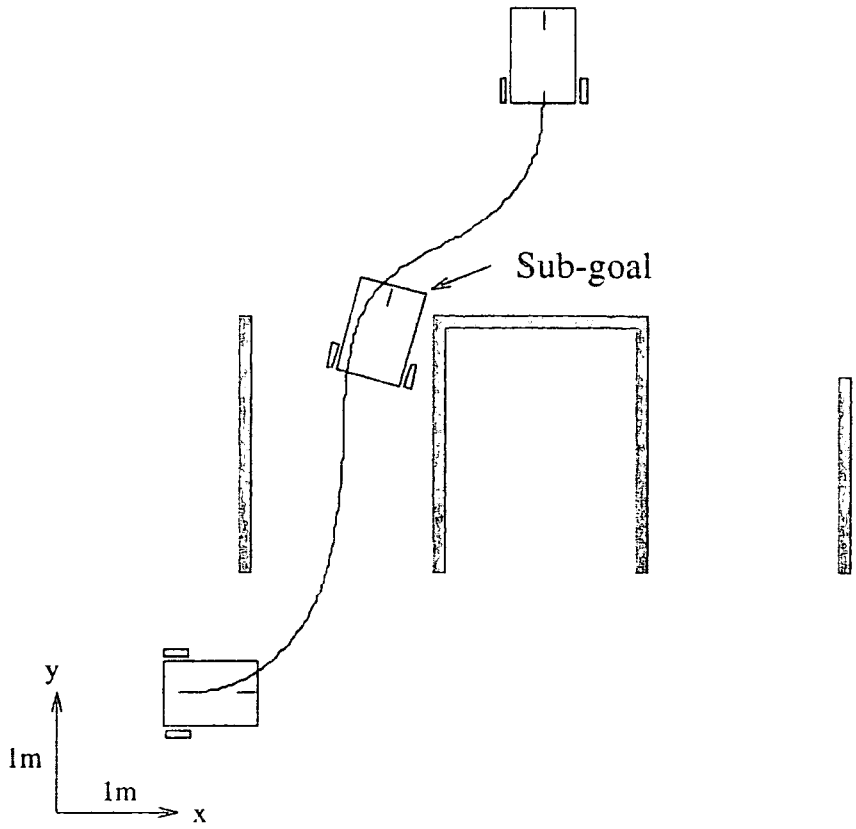

Figure 7.12: Vehicle Avoiding a Dead End Due to a Sub-goal.

Figure 7.12 shows how a sub-goal in the first corridor has prevented the vehicle from going up the central, blocked, corridor as it did in the example shown in Figure 7.8 and so is able to reach the goal.

7.5 Longer Laboratory Tests.

Previous tests using the vehicle have used occupancy grids which only included one or two obstacles to demonstrate the vehicle's avoiding action without complicating it by adding in the effects of many other objects. The tests presented here were carried out in the laboratory with the vehicle having a map with the desks, cupboards etc. included on the map in addition to some added objects (boxes). These tests have been recorded on video to provide a record of the vehicle in action.

Figure 7.13 shows the vehicle doing a circuit of the laboratory. By using the sub-goals shown the vehicle was made to follow a path around the edge of the laboratory before finishing close to its start point. The dotted line shows the path taken when no additional

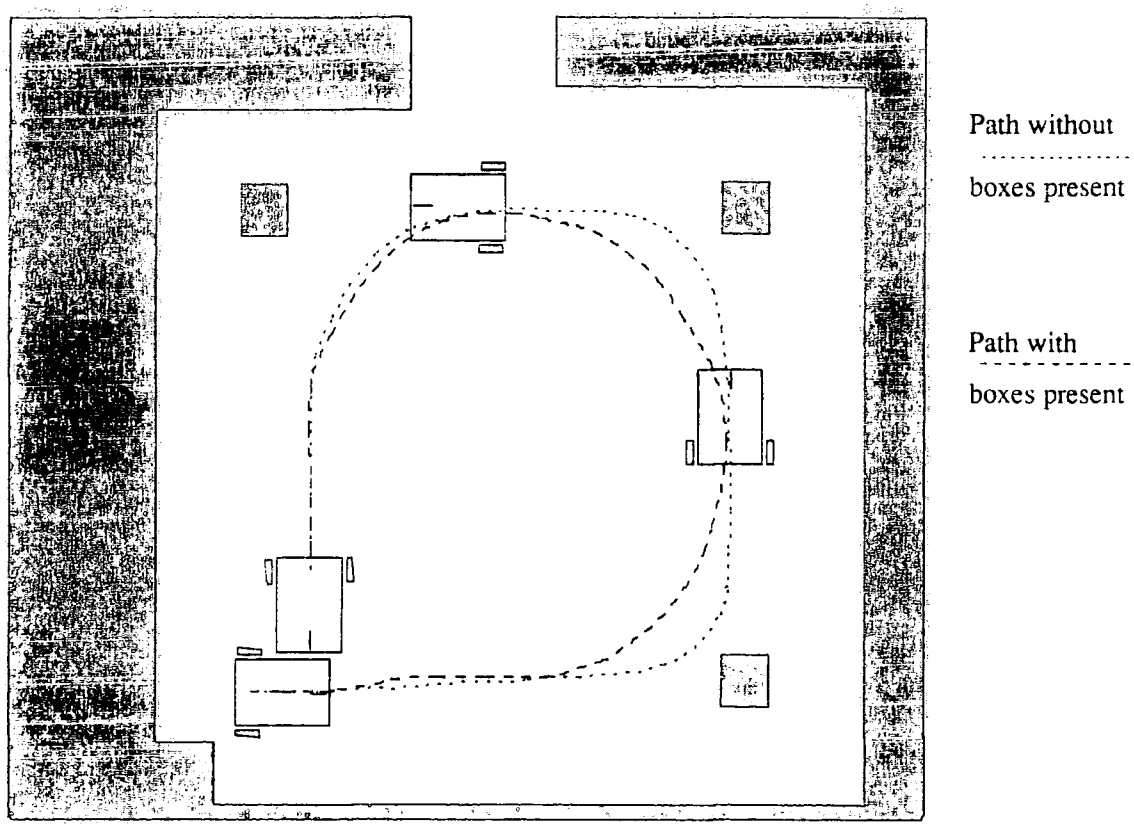

Figure 7.13: A Circuit of the Laboratory Avoiding Boxes.

objects were included while the dashed line shows how the presence of boxes at the corners of the laboratory altered the path by forcing the vehicle to go closer to the centre of the room.

If the boxes are moved closer into the centre of the room as is the case in Figure 7.14 it can be seen that the vehicle decides to go around the opposite side of two of the boxes in order to be better able to approach the sub-goals in between them.

In Figure 7.15 the vehicle is shown negotiating a corridor-like obstruction in the laboratory demonstrating how the control system can negotiate the vehicle around fairly large obstacles and through narrow gaps even without sub-goals.

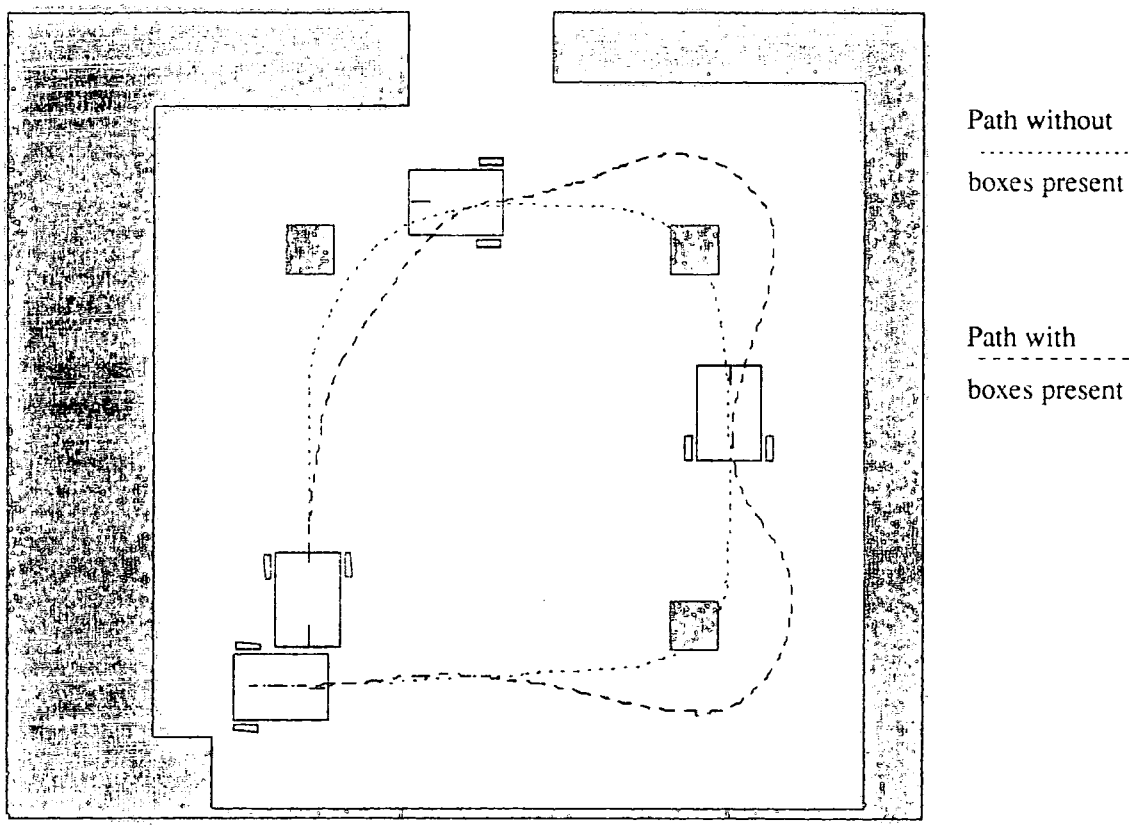

Figure 7.14: A Second Circuit of the Laboratory Avoiding Boxes.

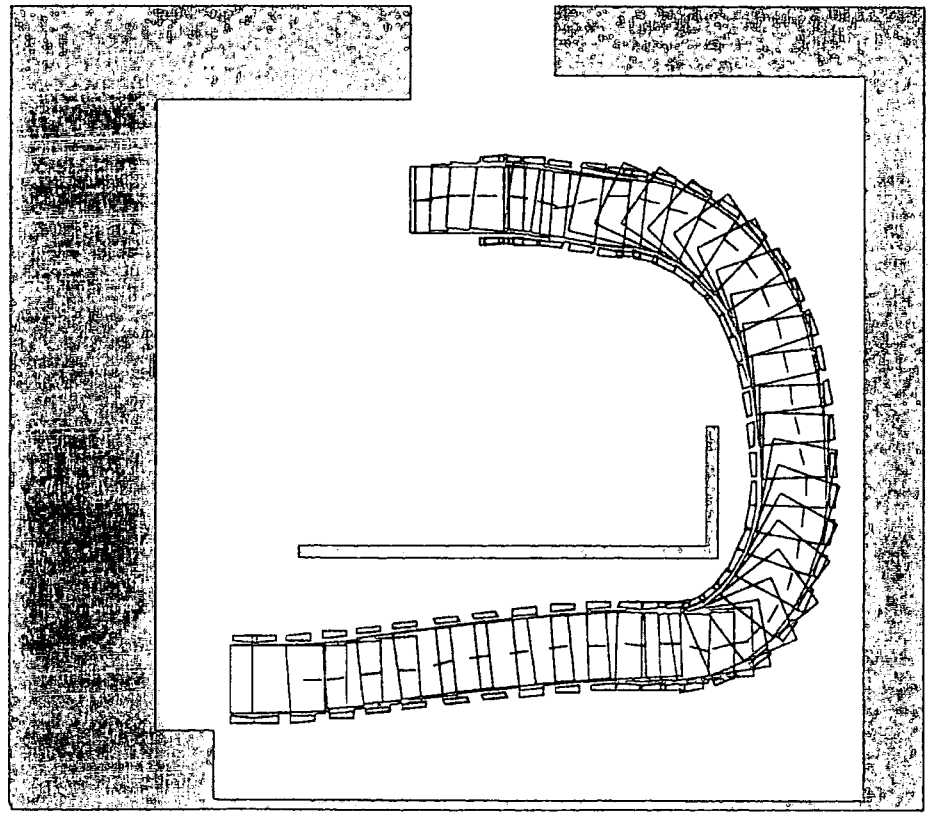

Figure 7.15: Negotiating a Corridor Like Object in the Laboratory.

Chapter 8

Conclusions and Suggestions for Further Work

8.1 Conclusions

This thesis has described the development of a fuzzy logic based control method for an automated guided vehicle. The work has centered on the development of a navigation system to guide the vehicle between goals, an obstacle avoidance system to prevent it hitting obstacles in its path and methods to sensibly combine these two systems. The navigation system has been realised as a conventional set of fuzzy rules while the obstacle avoidance function uses inhibitive rules. These rules were generated by considering the geometry of the vehicle and using this to examine important areas of an occupancy grid. Two new techniques, those of windowing and rule spreading, have been developed to combine the output of the navigation and obstacle avoidance systems. This has been shown to produce a successful blending of the sometimes conflicting goals of these systems to allow the vehicle to navigate to its goal while avoiding obstacles.

The resulting control system has been shown to be capable of guiding the vehicle without the formulation of a mathematical model or a plan. It has been shown to perform well even in the presence of noisy and inaccurate data from the vehicle's localisation system. It can therefore be concluded that the combination of conventional fuzzy logic techniques and the new techniques presented here provide powerful tools for the intelligent guidance of

automated vehicles.

The method adopted during this research of using a simulation program to allow testing of the theory behind the control system before using it in practice has been shown to give good results. The software used in the simulation could be directly transferred to the vehicle because it was written in the 'C' language and could be compiled for both the PC and transputers. This meant that many potential problems could be removed before the controller was tested on the vehicle. The more time consuming tests with the real vehicle could then be used to show that the control method could cope with sensor noise and position uncertainty.

The importance of this was demonstrated in Chapter 4 which showed how the use of the simulator to assess performance throughout the design stage enabled the rapid development of a navigation controller. Tests with the real vehicle were necessary to assess the accuracy of the limits placed on the rates of change of velocity and steering angle. This allowed the development of a fuzzy logic based velocity controller to optimise the controller's performance. The simulation also played a vital role in showing that some navigation problems are caused by inherent conflicts between rules. Being able to carry out simulated tests meant that it could be demonstrated that the techniques used for combination with the obstacle avoidance system alleviated the conflicting rule problem without endangering the vehicle.

The use of a reactive control method rather than relying on planning has been supported by other researchers on the grounds that such methods are more readily able to cope with changes in, or uncertainty about, the environment. The results presented in this thesis support this argument by demonstrating how well the controller performs in the presence of inaccurate location data. Simulation results presented at the end of Chapter 4 demonstrated that the fuzzy logic control system operated well in the presence of noise and this has been borne out by the successful operation of the controller on the real vehicle despite the uncertain accuracy of the localisation system. It was, however, noted in Chapter 7 that there are some situations where reactive control is too short-sighted to provide good guidance and the necessity of additional guidance, possibly by the provision of sub-goals, by a higher level in the control hierarchy was recognised. Some results from tests using simple sub-goals have demonstrated how this additional information can improve the performance of the control system in difficult situations.

As was stressed in Chapter 1, one of the main aims of the research was to examine the way in which two systems, one designed to guide a vehicle to a specified position and another to avoid obstacles, could be combined. Consideration of the problem lead to the conclusion that the first priority of a guidance system must be to prevent the vehicle striking an obstacle. An important second priority, however, is that the vehicle should avoid an obstacle in a manner consistent with achieving its goal. The basic avoidance method introduced in Chapter 5 has been shown to comply with these principles. The method depends on avoidance sets linked to areas on the occupancy grid containing mapping information about the environment. Particular areas are linked to particular steering angle sets and inhibit the use of that set in the output. The degree of inhibition depends on how close the area is to the vehicle. These techniques enable a mask vector to be produced, the entries in which represent the desirability of the different steering directions.

The inhibitive obstacle avoidance method prevents the vehicle colliding with obstacles and is combined with the information from the navigation system to provide guidance on the best path to the goal. By combining the fuzzy outputs of the two systems the information about preferred steering angles takes into account the demands of both obstacle avoidance and goal seeking. Unfortunately conventional defuzzification techniques can misinterpret the combined fit vector and yield an output likely to produce a collision. It was therefore found necessary to develop a windowing technique which confines the defuzzification stage to those sets neighbouring the most activated set. This technique has been shown to remove the indecision present in conventional fuzzy controllers caused by inherent conflicts in the rules or the presence of obstacles. A modification to this basic technique, dynamic windowing, was introduced in Chapter 6 as a method of smoothing transitions between outputs in the controller and reducing the need for drastic changes in velocity to be demanded by the velocity controller.

A second new technique, rule spreading, was introduced to cope with the situation when all the sets activated by the navigation system were inhibited by the avoidance system. Rule spreading uses a normal distribution to add activation to output sets depending on their proximity to sets activated by the navigation controller, thus ensuring that all sets are activated to some extent. Experimental tests presented in Chapter 6 identified that large obstacles posed a particular problem for the controller and showed that increasing the consideration given to alternative paths in the presence of large obstacles by the use of

variable rule spreading improved the controller's performance. Furthermore it was found that in cases where there was no clear choice as to which of two directions was the best the controller could enter an oscillatory state. From this it was concluded that two new rules should be introduced to encourage the controller to prefer steering angles close to those it had suggested on the previous interrupt or were close to the present steering angle.

While several reactive guidance schemes have been described by other researchers for freely rotating cylindrical vehicles, the aim of this research was to produce a method for a vehicle with a more conventional shape and steering mechanism. This has been achieved by considering the vehicle's geometry and steering capabilities when generating obstacle avoidance sets as described in Chapter 7. This generation technique means that the avoidance method could be applied to many different types of vehicle.

The effect that changing the parameters used during the generation of avoidance sets had on the performance of the research vehicle was investigated through a series of tests described in Chapter 7. These led to the conclusion that for the research vehicle the set areas should extend up to two metres in front of the vehicle and one metre to the side. Those definitions dealing with steering straight ahead should be kept narrow to allow the vehicle to negotiate small gaps, while the sets for larger steering angles should be fanned out more widely to discourage the controller from turning the vehicle towards potential obstacles.

Through the numerous results presented in Chapter 7 it has been shown that the controllers and techniques described in this thesis can be combined to give a powerful guidance system capable of negotiating many obstacles impeding the path of the vehicle. By the use of a real vehicle with an imprecise localisation system and by simulating errors in position it has been shown that the techniques provide guidance of the vehicle even in the presence of noise. While the navigation controller was designed for use in relatively uncluttered environments, examples of the placements of sub-goals have shown how the vehicle's performance could be further improved by guidance from higher levels of a control hierarchy. As well as demonstrating how fuzzy logic techniques can be successfully used to provide a fairly sophisticated vehicle controller this thesis has introduced techniques which would be of use in any fuzzy logic control system where a method of resolving conflicts between rules is needed.

8.2 The Contribution of the Presented Work

The aim of the work presented in this thesis was to explore ways in which the strengths of fuzzy logic control could be exploited for the low level control of an automated guided vehicle. The main contributions of the work can be described as follows.

1. The concept of combining positive and inhibitive fuzzy rules by multiplying the fuzzy fit vector by an inhibitive mask vector.

2. The use of a windowing stage before defuzzification to prevent the adverse effects of conflicting rules in a fuzzy controller. Such conflicts can arise from within a conventional fuzzy logic controller or from the use of inhibitive rules.

3. The use of dynamic windowing to produce a smoother dynamic response.

4. The use of a normal distribution to spread the influence of rules throughout the fuzzy fit vector to provide alternatives to output sets which may be inhibited.

5. The variation of the rule spreading constant to improve obstacle avoidance performance.

6. The use of areas of an occupancy grid linked to output sets to produce inhibitive obstacle avoidance rules.

7. The development of an automatic method for generation of avoidance set areas from consideration of the size and shape of a vehicle and its steering mechanism.

8.3 Suggestions for Further Work

There are many issues raised by the work described in this thesis which are worthy of further investigation but which could not be pursued within the time available. The proposals for further work cover improvements and additions to the research vehicle, modifications to the systems outlined in this thesis and work on the systems, such as a high level planning stage, which have not been considered but which are essential for a complete automated vehicle control scheme.

For the vehicle itself the first priority must be to provide it with sensor systems to detect the presence of objects, thus enabling the vehicle to modify the occupancy grid during operation. Some work was undertaken in this direction to prepare Polaroid ultrasonic sensors to be fitted on the vehicle [53] and other work on sensors is currently being carried out at Durham. The use of sensor derived information to fill the certainty grid would enable the performance of the fuzzy logic controller with probabilistic obstacle information to be properly assessed. It is believed that the obstacle avoidance system presented in this thesis would be able to operate on such data with little difficulty and this was one of the stated reasons for adopting the occupancy grid approach. In conjunction with this it would be useful to extend the simulation to include the sensor acquisition of data. If the sensors used could be characterised in a manner similar to that done by Elfes [18] it would be possible to model the way in which occupancy grids are built up to assess in simulation the way this affects obstacle avoidance.

Further improvements in the localisation system would also be valuable. The present system could be improved by the inclusion of more advanced methods of reducing the errors in the position estimates. This could take the form of expanding odometric error bands as used by Pears [16] or by the use of Kalman filtering techniques [21]. If funds permitted it would be valuable to have a much more accurate and sophisticated localisation system such as that used by the GEC Caterpillar based on a laser range-finder and barcodes and available commercially. In addition to this it would be useful to be able to power the onboard computer from a battery to dispense with the mains cable and to replace the current wire link to the host with a radio link which would enable monitoring of the vehicle during a test.

If the vehicle were to be tested using sensor acquired information about obstacles it is anticipated that a mapping between occupancy values and inhibition values more complicated than the one currently in use (Equation 5.3) would give better results. Depending on the characterisation of the sensors it might well be more appropriate to replace this linear equation with a curve. This curve might further reduce the inhibitive effect of areas with low occupancy values, to allow for the acquisition of more information by sensors, and give more of an inhibitive effect to high occupancy values. A curve with such properties is the sigmoidal function much used in neural networks. It might well be useful to include the ability to vary the characteristics of the curve depending on the density of the occupancy

grid. In areas of high object density it is worthwhile investigating objects more closely than in fairly clear areas where the vehicle will not have much difficulty finding a completely free path.

Another capability which could be included in the controller is that of reversing. This ability at first appears to be trivial but upon investigation it can be seen that without guidance from higher levels in the control hierarchy it cannot be easily included in the control scheme. Physically the research vehicle is already capable of reversing, and by the addition of sign changes the existing navigation controller can be, and has been, made to control the vehicle in reverse. To extend the obstacle avoidance system is also trivial. A new set of avoidance sets would be necessary because the vehicle has different steering characteristics in reverse. These avoidance sets could easily be generated using the current program to allow for the fact that the rear of the vehicle would swing left during right turns and vice versa. The problem occurs, however, when considering when to change direction. The addition of a direction change greatly increases the possibility that a vehicle may become locked within a dynamic loop. As a simple illustration of this if a vehicle is faced with a wall on the direct path between it and the goal it might drive up to the wall, perceive there is no gap, reverse, then decide that the best route to the goal lay straight ahead, drive up to the wall and continue in a similar manner to a fly faced with a window. Clearly some guidance is needed from the path planning level which can recognise such problems and force the vehicle to go around the wall. Reversing has to be considered as one of those situations which cannot be left to a reactive controller alone and must be guided. The present controller would be quite capable of implementing reversing manoeuvres such as a three point turn given suitable guidance from the planning layer.

At the present time the docking behaviour of the vehicle is accurate only to within five or ten centimeters and a few degrees. It could be substantially improved by replacing the fuzzy logic controller for the final stage by a PID controller. Once the fuzzy logic control has brought the vehicle into a position where it need only follow a simple curve to reach the goal the path is easy to define mathematically. If such a method were used the fuzzy logic controller might be able to be greatly simplified by the removal of a whole section of rules concerned with final docking. One of the reasons for not implementing this, however, is that no docking controller can ever be more accurate than the localisation system which provides its information.

It might be useful for higher layers of the control hierarchy to be able to modify the parameters and rules used by the controllers. Learning algorithms such as neural networks and genetic algorithms have been proposed for developing and fine tuning fuzzy logic controllers. The set definitions for what is considered a long, medium or short distance might benefit from being modified depending on the vehicle's velocity. Similarly the definitions of obstacle avoidance sets could be varied with velocity and with the degree of uncertainty about the vehicle's position. Different fuzzy navigation controllers designed for different situations such as moving down corridors, crossing doorways or exploring for map building purposes could be applied instead of the present controller, or blended together as proposed by Saffiotti et al [47]

Another more obvious task which higher layers of the control hierarchy need to be able to perform is the assessment of mapping information to provide guidance on the long term path to be followed by the vehicle. These layers need to be able to perform the sort of tasks which a human driver would perform when planning a route from a map, or be able to use information from the environment (such as signs) to guide the vehicle. There is still much work to be done in this field, particularly in examining the interface between planning and reaction. Considerable work needs to be done on how much or how little information needs to be transferred between layers in a hierarchical control system to yield optimum performance but the work presented here has shown that fuzzy logic techniques can allow profitable cooperation between systems without the need for excessive communication.

References

[1] L.A.Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[2] *Ultrasonic Ranging System.* Cambridge, Mass.: Polaroid Corporation, 1982.

[3] H.R. Everett, "A Multi Element Ultrasonic Ranging Array," *Robotics Age*, July 1985.

[4] P.K.C. Hande and P.C. Sharma, "A Fully Compensated Digital Ultrasonic Sensor for Distance Measurement," *IEEE Transactions on Instrumentation and Measurement*, vol. 30, June 1984.

[5] R. Kuc, "A Spatial Sampling Criterion for Sonar Obstacle Avoidance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 686–690, 1990.

[6] Anita M. Flynn, "Combining Sonar and Infra Red Sensors for Mobile Robot Navigation," *International Journal of Robotics*, vol. 7, pp. 5–14, December 1988.

[7] B. Steer and T. Atherton, "Design for Navigation," in *1990 IEEE International Conference on Robots and Automation*, 1990.

[8] B. Barshan and R. Kuc, "Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 560–569, 1990.

[9] R. Kuc and V.B. Viard, "A Physically Based Navigation Strategy for Sonar Guided Vehicles," *International Journal of Robotics*, vol. 10, pp. 75–87, April 1991.

[10] B. Barshan and R. Kuc, "A Bat-Like Sonar System for Obstacle Location," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 4, pp. 636–646, 1992.

[11] L. Vietze and I. Hartman, "An Ultrasonic Phased Array Sensor for Robot Environment Modelling and Fast Detection of Collision Possibility," in *Infronation and Control Problems in Manufacturing Technology. IFIP/IFOR/IMACS Symposium, Madrid 89*, 1989.

[12] W.S.H. Munro *et al.*, "Ultrasonic Vehicle Guidance Transducer," *Ultrasonics*, vol. 26, pp. 350–354, November 1990.

[13] Huosheng Hu, "Distributed Architecture for Sensing and Control of a Mobile Robot," Tech. Rep. OUEL 1836/90, Oxford University, 1990.

[14] C. Frohlich, F. Freberger, and G. Schmidt, "A 3D Laser Range Camera for Sensing the Environment of a Mobile Robot.," *Sensors and Actuators A-Physical*, vol. 26, pp. 453–458, 1991.

[15] Ingemar J. Cox, "Blanche – An Experiment in Guidance and Navigation of an Automonous Robot Vehicle," *IEEE Transactions on Robotics and Automation*, vol. 7, April 1991.

[16] N.E. Pears, *Low Level Guidance of an Experimental Autonomous Vehicle.* PhD thesis, Durham University, 1989.

[17] A. Bradshaw, "Sensory Systems for Robotic Control," *Transactions of the Institute of Measurement and Control*, vol. 23, March 1990.

[18] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, vol. 22, no. Issue 6, pp. 46–57, 1989.

[19] M.H. Soldo, "Fusion Without Representation," *SPIE, Sensor Fusion II*, vol. 1198, 1989.

[20] N.E. Pears and J.R. Bumby, "Guidance of an Automated Guided Vehicle Using Low Level Ultrasonic and Odometry Sensor Systems," *Transactions of the Institute of Measurement and Control*, vol. 11, no. 5, pp. 231–248, 1989.

[21] J.J. Leonard and H.F. Durrant Whyte, "Mobile Robot Localisation by Tracking Geometric Beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, April 1991.

[22] L. Tarassenko, M.J. Brownlow, and A.F. Murray, "VLSI Neural Networks for Autonomous Robot Navigation," in *International Neural Networks Conference 1990 (Paris)*, July 1990.

[23] R.A. Brooks, "Intelligence Without Representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[24] P. Hoppen, T. Knieriemen, and E von Puttkamer, "Laser Radar Based Mapping," in *1990 IEEE International Conference on Robots and Automation*, 1990.

[25] Dong Woo Choo, "Certainty Grid Representation for Robot Navigation by a Bayesian Method," *Robotica*, vol. 8, no. 2, pp. 159–165, 1990.

[26] J. Borenstein and Y. Koren, "Histogramic in Motion Mapping for Mobile Robot Obstacle Avoidance," *IEEE Transactions on Robotics and Automation*, vol. 7, August 1991.

[27] A. Shaout and C. Isik, "Mobile Robot Map Updating Using Attributed Fuzzy Tournament Matching," *SPIE - Intelligent Robots and Computer Vision III : Algorithms and Techniques*, vol. 1192, 1989.

[28] C.L. Shih, T.T. Lee, and W.A. Gruver, "A Unified Approach to Robot Motion Planning with Moving Obstacles," *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, no. 4, pp. 903–915, 1990.

[29] J.F. Delamadrid and M.L. Gini, "Path Tracking Through Uncharted Moving Obstacles," *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, no. 6, pp. 1408–1422, 1990.

[30] D. Feng and B.H. Krogh, "Satisficing Feedback Strategies for Local Navigation of Autonomous Mobile Robots," *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, no. 6, pp. 1383–1395, 1990.

[31] V. Lumelsky and T. Skewis, "Incorporating Range Measurement in the Robot Navigation Function," *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, no. 5, pp. 1058–1069, 1990.

[32] J. L. Jones and A. M. Flynn, *Mobile Robots: Inspirations to Implementation.* Wellesley, Massachusetts: A. K. Peters, 1993. ISBN 1-56881-011-3.

[33] J. Borenstein and Y. Koren, "Real Time Obstacle Avoidance for Fast Mobile Robots," *IEEE Transactions on Systems Man and Cybernetics*, vol. 19, no. 6, pp. 1179–1187, 1989.

[34] R.C. Arkin, "Motor Schema-Based Mobile Robot Navigation," *International Journal of Robotics*, vol. 8, no. 2, pp. 92–112, 1989.

[35] M.G. Slack, "Navigation Templates: Mediating Qualitative Guidance and Quantative Control in Mobile Robots," *IEEE Transactions on Systems Man and Cybernetics*, vol. 23, no. 2, pp. 452–466, 1993.

[36] J. Borenstein and Y. Koren, "Teleautonomous Guidance for Mobile Robots," *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, no. Issue 6, pp. 1437–1443, 1990.

[37] D.W. Payton, J.K. Rosenblatt, and D.M. Keirsey, "Plan Guided Reaction," *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, no. 6, pp. 1370–1382, 1990.

[38] Y.F. Li and C.C. Lau, "Development of Fuzzy Algorithms for Servo Systems," *IEEE Control Systems Magazine*, 1989.

[39] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewoods Cliffs, New Jersey: Prentice Hall, 1991. ISBN 0–13–612334–1.

[40] M. Maeda, Y. Maeda, and S. Murakani, "Fuzzy Drive Control of an Autonomous Mobile Robot," *Fuzzy Sets and Systems*, vol. 39, pp. 195–204, January 1991.

[41] R. Garcia Rosa, D. Kumpel, and M.C. Garcia Alegre, "A Local Navigation System Based on Fuzzy Control," in *Proceedings of the 5th CIM Europe Conference*, (Brussels - Luxembourg), pp. 379–388, ECSC-ZAEC, May 1989. ISBN 1–85423–040–9.

[42] J. Yen and N. Pfluger, "A Fuzzy Logic Based Robot Navigation System," in *Proceedings of the AAAI Fall Symposium on Mobile Robot Navigation*, pp. 195–199, 1992.

[43] A. Saffiotti, E.H. Ruspini, and K.G. Konolige, "A Fuzzy Controller for FLAKEY, an Autonomous Mobile Robot," Tech. Rep. 529, SRI International, March 1993.

[44] Yu-Lan Zhou and Zhengyue Qui, "An Expert Self Learning Fuzzy Controller," in *IFAC, Artificial Intelligence in Real Time Control 89*, pp. 69–73, 1989.

[45] D.T. Pham and D. Karaboga, "A New Method to Obtain the Relation Matrix for Fuzzy Logic Controllers," in *Applications of Artificial Intelligence in Engineering VI*, (Southampton S04 2AA), pp. 567–581, Computational Mechanics Publications Ltd, 1991. ISBN 1 85312 141 X.

[46] R.C. Arkin and R.R. Murphy, "Autonomous Navigation in a Manufacturing Environment," *IEEE Transactions on Robotics and Automation*, vol. 6, no. No.4, pp. 445–54, 1990.

[47] A. Saffiotti, K.G. Konolige, and E.H. Ruspini, "A Multivalued Logic Approach to Integrating Planning and Control," Tech. Rep. 533, SRI International, July 1993.

[48] J.W. Baxter, "Altera Text Design Files used by the AGV."
Reference CODE/AGV/JWB/03, September 1994.

[49] J.W. Baxter, "Controlling Software for an AGV Using Fuzzy Logic for Navigation and Obstacle Avoidance, Source Code for Program Fuzzyrun.C." Reference CODE/AGV/JWB/02, August 1994.

[50] J.W. Baxter, "Simulation of an AGV Using Fuzzy Logic for Navigation and Obstacle Avoidance, Source Code for Program OBAGV5.C." Reference CODE/AGV/JWB/01, August 1994.

[51] J.W. Baxter and J.R. Bumby, "Fuzzy Logic Guidance and Obstacle Avoidance Algorithms for Autonomous Vehicle Control," in *Proceedings of the 1st IFAC International Workshop on Intelligent Autonomous Vehicles*, pp. 259–264, Pergamon Press, April 1993.

[52] J.W. Baxter and J.R. Bumby, "Modifying the Fuzzy Fit Vector to Avoid Unwanted Outputs, With Particular Reference to Autonomous Vehicle Control." Colloquium on Neural Networks and Fuzzy Logic in Measurement and Control, Liverpool, March 1993.

[53] J.W. Baxter, "Autonomous Guided Vehicles - A Report on Year One."
Reference REV/AGV/JWB/02, October 1992.

[54] H.J Zimmerman, *Fuzzy set theory and its applications.* P.O. Box 322 3300 AH Dordrecht, THE NETHERLANDS: Kluwer Academic Publishers, 1991. 2nd Edition.

[55] M. Mizumoto and H-J Zimmermann, "Comparison of Fuzzy Reasoning Methods," *Fuzzy Sets and Systems*, vol. 8, pp. 253–283, 1982.

[56] M. Mizumoto, "Fuzzy Controls Under Various Fuzzy Reasoning Methods," *Information Sciences*, vol. 45, pp. 129–151, 1988.

[57] M. Mizumoto, "Improvement Methods of Fuzzy Controls," in *Proceedings of the 3rd IFSA Congress*, pp. 60–62, 1988.

[58] J.W. Baxter, "Miscellaneous Software Used During AGV Research."
Reference CODE/AGV/JWB/04, September 1994.

[59] J.W. Baxter, "Hardware Interconnections on the AGV."
Reference CON/AGV/JWB/01, September 1994.

Appendix A

Fuzzy Logic Control

The basis behind fuzzy logic control is the fuzzy set theory pioneered by Zadeh. Detailed and rigorous introductions to the fundamental theory can be found in the books by Kosko[39] and Zimmerman[54]. The following is a brief introduction to the field.

Fuzzy set theory is a system of multi-valued logic based on assigning values to the *degree* of membership of a set. The maximum membership of a set is usually 1, is a full member, and the minimum 0, is not a member, with values in between indicating partial membership. As an example for a set of white objects something which is a light grey might have a membership of 0.8 and a dark grey object a membership of 0.3. For use in control the sets are defined to divide the state variables up into useful sized sets and the membership values of the inputs in these sets are used to implement the control strategy. These sets are often given names and referred to as *linguistic variables.* As an example a distance of 10m might be considered 'LARGE' but it also could be considered 'MEDIUM'. In fuzzy set theory this could be represented by saying that the membership of the distance 10m in the set 'LARGE' is 0.9 and its membership of the fuzzy set 'MEDIUM' is 0.5.

The dividing of state variables into linguistic sets enables a control strategy to be made up of rules which are understood and can be derived by human operators without the use of a mathematical model. This enables processes which are hard to model, or for which no precise mathematical data can be obtained, to be controlled. Control rules take the form of simple IF THEN rules such as :-

IF distance to goal is 'LARGE' and speed is 'LOW'
then acceleration is 'POSITIVE LARGE'

or

IF distance to goal is 'SMALL' and speed is 'MEDIUM'
then acceleration is 'NEGATIVE MEDIUM'

The translation of this type of rule into a control algorithm can be done by many different but similar methods. The main points are how a fuzzy set is defined, how the AND and OR operators are defined and how the inference process is carried out. The inference process is also linked to the *defuzzification* process which takes all the output set activations and reduces it to a single, crisp, output for the controller. The methods and terminology used here are those used by Kosko [39].

A.1 Numerical Fuzzy Sets

A fuzzy set is defined over a *universe of discourse* which covers all the values which it is possible for that variable to take. The nature of a fuzzy set is best shown graphically and can be seen in Figure A.1 where three of the most common shapes for fuzzy sets are shown. These are triangular, trapezoidal and bell shaped sets. They are often chosen to be symmetric and therefore easily characterised by their centroid where the membership value in the set is usually one. The triangular and trapezoidal sets have the advantage of being easy to code and store while the bell shaped sets are more complex, based on an exponential function, but have the property that they have non zero membership for the whole universe of discourse.

Set A might be a set of numbers very close to three in which the number 1.0 has membership 0.3. Set B could represent numbers around eleven where 8 has membership 0.5 and 12 membership 1. The third set might be of numbers near to 20.

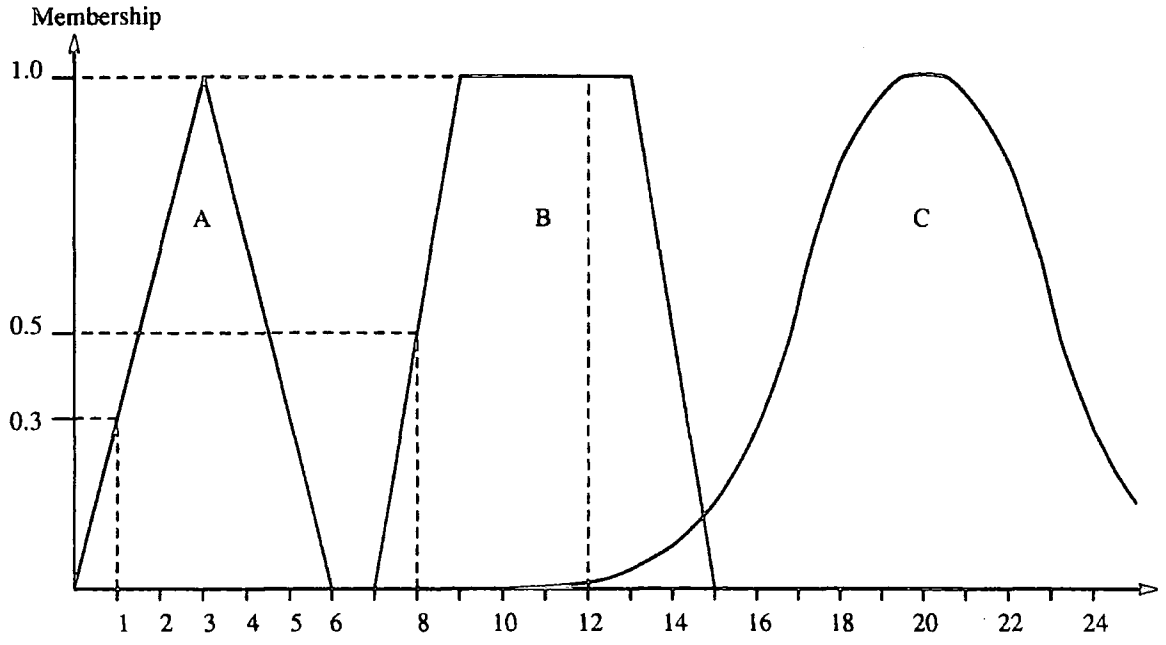

Figure A.1: Three Common Shapes for Fuzzy Sets.

A.2 Fuzzy Operators

The most important operators for fuzzy logic control are AND, OR and NOT. Under the most common scheme used AND is carried out by taking the minimum membership value indicated by the sets while OR is carried out by taking the maximum. NOT is simply 1-membership value. This can be illustrated by considering Figure A.2 which shows two sets 'SMALL' and 'MEDIUM'. If the function $u(A, x)$ gives the membership of x in the set A then from the diagram it can be seen that $u(\text{SMALL}, 4.5) = 0.75$ and $u(\text{MEDIUM}, 4.5) = 0.5$. Using this notation the AND operator is defined by Equation A.1

$$u(A \text{ AND } B, x) = \min(u(A, x), u(B, x)) \tag{A.1}$$

and therefore in the example

$$\begin{aligned} u(\text{SMALL AND MEDIUM}, 4.5) &= \min(0.75, 0.5) \\ &= 0.5 \end{aligned}$$

Similarly the OR operator is defined by Equation A.2

$$u(A \text{ OR } B, x) = \max(u(A, x), u(B, x)) \tag{A.2}$$

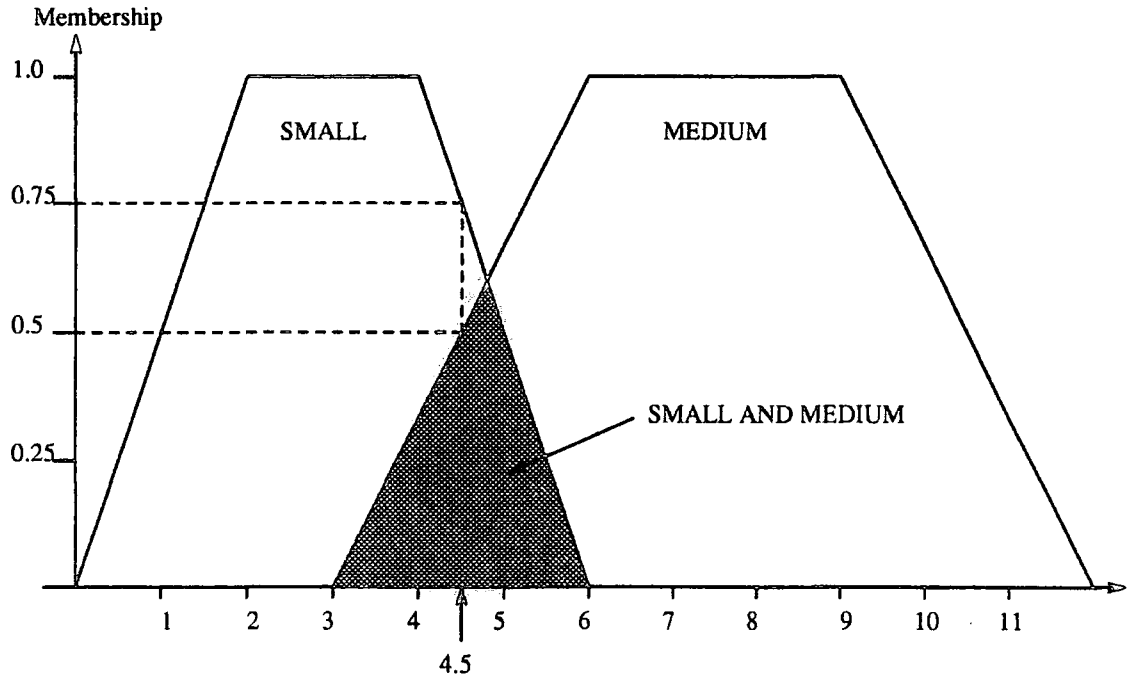

Figure A.2: Two Overlapping Fuzzy Sets

and therefore in the example

$$\begin{aligned} u(\text{SMALL OR MEDIUM}, 4.5) &= \max(0.75, 0.5) \\ &= 0.75 \end{aligned}$$

and finally the NOT operator is defined by Equation A.3.

$$u(\text{NOT } A, x) = 1 - u(A, x) \tag{A.3}$$

A.3 Correlation

Correlation is the THEN statement in fuzzy logic which links sets together. For control purposes this is the linkage between the input state variables and the outputs. In the terminology of Kosko the links are known as *Fuzzy Associative Memories* or FAMs and each FAM represents a single rule. Fuzzy sets are defined to cover all possible outputs in the same way that they are defined for the inputs. The link between the input sets and the output sets is made by taking the membership value that the input sets have in the rule and using this as the *activation* of the output set.

For example using the rule

IF distance to goal is 'LARGE' and speed is 'LOW'
then acceleration is 'POSITIVE LARGE'

If the inputs to the controller were such that distance had membership of 0.3 in the set 'LARGE' and speed had membership 0.7 in the set 'LOW' then using the minimum operator as the fuzzy AND the activation of the rule would be 0.3 and the output set for a 'POSITIVE LARGE' acceleration would have an activation of 0.3.

For a controller there are usually many such rules giving rise to a large number of FAMs. The FAMs can be grouped together in a *FAM bank matrix* which is a convenient way of displaying the rules. The matrix has as many rows as there are sets of one input variable and as many columns as there are sets of a second input variable. The matrix entries then show the consequent output sets for all possible combinations of those two input variables. There may not be an entry for every combination indicating that there is no rule for such a possibility. One of the advantages of having highly overlapping input sets is that these blank entries in the matrix are covered by adjacent rules.

There are numerous ways of interpreting the activation value of the output set and a detailed discussion can be found in [55] and [56]. There are two main methods however, they are called the *correlation minimum* and *correlation product* methods and are shown graphically in Figure A.3. Correlation minimum clips the top off the output set by limiting the maximum membership values in the set to the activation level. Correlation product shrinks the whole set by multiplying all the membership values in the output set by the activation level.

In a large system with many rules an output set could be activated more than once. In this case the sum of all the activations are built up into a *fuzzy fit vector* which contains the sum of all the activations for each set. It is also possible to allow a weight to be applied to each rule if desired to indicate its importance relative to the other rules. The result of applying all the rules gives several activated sets covering the output universe of discourse. In order for a control action to take place the sets must be *defuzzified* to give a single output for the control action.

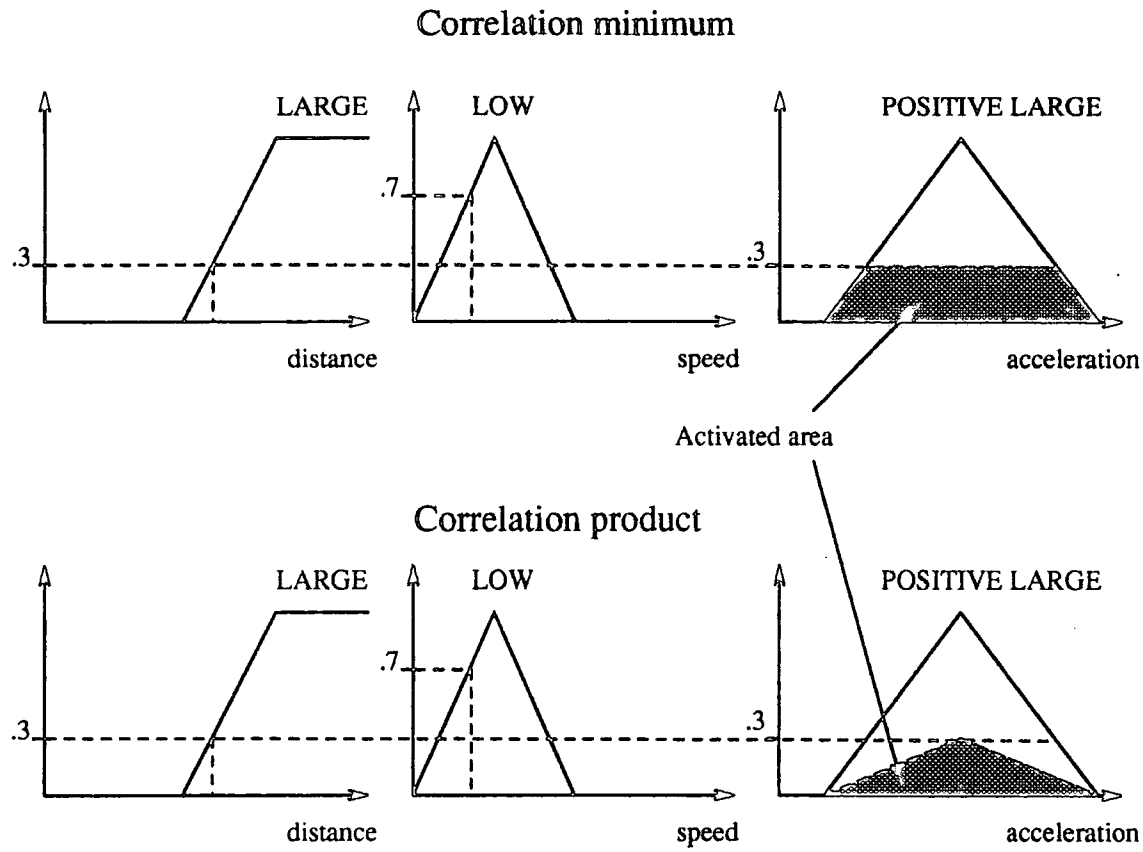

Figure A.3: The Two Different Correlation Methods.

A.4 Defuzzification

There are many different possible defuzzification methods, some of the merits of which are discussed by Mizzumoto [57]. One of the most common defuzzification routines is the *centre of gravity* method. This finds the crisp output value by taking a weighted average of the centroids of the output sets as shown in Equation A.4.

$$\text{Output Centroid} = \frac{\sum_{i=1}^{n} c_i a_i}{\sum_{i=1}^{n} a_i} \tag{A.4}$$

where a_i = Activated area of the ith fuzzy output set.

c_i = Centroid of the ith fuzzy output set.

The weights used are the activated areas of the sets found by one of the two correlation methods. For symmetric sets the centroid never moves regardless of the activation which is a useful property since the centroid need only be found once rather than being recalculated for different activation values. The correlation product method has the additional advantage

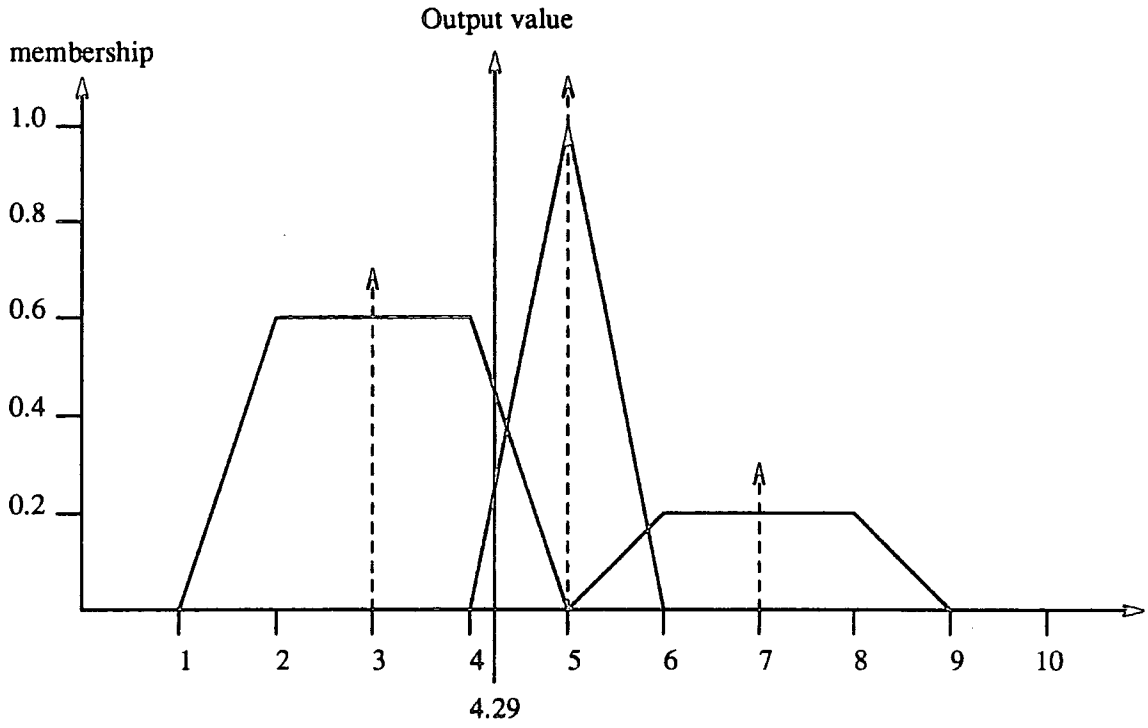

Figure A.4: Defuzzification of the Output

that the centroids of non-symmetric sets also remain constant and the activated area of a set is directly proportional to its activation value. This means that sets can be entirely characterised by their centroids and gives the correlation product method a computational advantage over correlation minimum.

Figure A.4 shows the defuzzification of three output sets one with its centroid at 3 and activated to degree 0.6, the next centered around 5 with an activation of 1.0 and the third with an activation of 0.2 and centered around 7. This gives the three sets areas of 1.8,1 and 0.6 respectively so the defuzzified output can be calculated as being :-

$$\frac{3 \times 1.8 + 5 \times 1.0 + 7 \times 0.6}{1.8 + 1.0 + 0.6} = 4.29 \text{ (2d.p.)}$$

A.5 The Structure of a Fuzzy Logic Controller

A fuzzy logic controller uses the methods described in the previous sections to find the required control output from some input states by applying fuzzy rules. A simple block

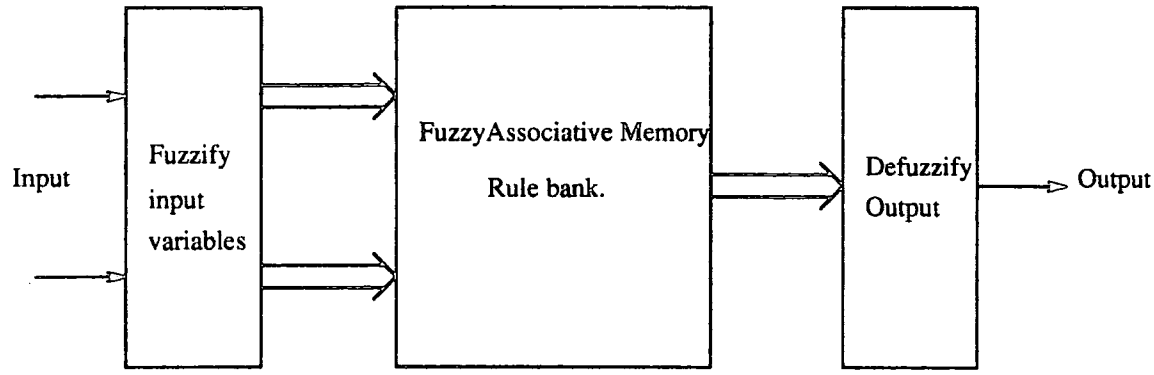

Figure A.5: Block Diagram of a Simple Fuzzy Logic Controller.

diagram of a controller can be seen in Figure A.5. The input variables are fuzzified and the membership values of the current inputs in the input fuzzy sets are used by the FAM bank matrix to derive the activation of the output fuzzy sets. This *fuzzy fit vector* is passed to the defuzzifier which produces an output which is used to control the system.

From the point of view of implementing a controller the fuzzifier uses set definitions to decide the membership of the input values in the input sets. Each pair of sets (in the two input case) has a corresponding entry in the FAM bank matrix which indicate which output set the rule is associated with. The correlation between the two input sets decides the activation of the output set and its entry in the fuzzy fit vector is modified. When all the rules have been applied the fuzzy fit vector is defuzzified and the result output as the control value.

Appendix B

Beacon Location System

This appendix describes the way in which the beacon location system operates to provide information about the location of the vehicle while it is moving. The system has already been outlined in Section 2.6.2 but this appendix provides more information about the way in which the beacon information is gathered and processed.

As has been explained in Chapter 2 the odometry system keeps track of the position of the vehicle in the short term but errors in measurement caused by wheel slip and uneven surfaces grow over time. The purpose of the beacon location system therefore is to provide an estimate of the vehicle's position which can correct these errors over larger distances. The system had to be cheap and while not particularly accurate able to provide an upper bound for the errors in locating the vehicle. The system tries to find the vehicle's x,y position and its orientation θ, measured as an angle from the y axis as shown in Figure B.1. All measurements are taken to the midpoint between the two rear driving wheels and are relative to one corner of the laboratory. By triangulating from the angles measured to the beacons it is possible to locate the vehicle. Section B.1 introduces the theory of triangulation and discusses the methods used to cope with errors present in the measurement of the angles to the beacons. Sections B.2 and B.3 describe the hardware used to sense the beacons while the functions performed in software to control individual scanners and the location system as a whole are described in Section B.4.

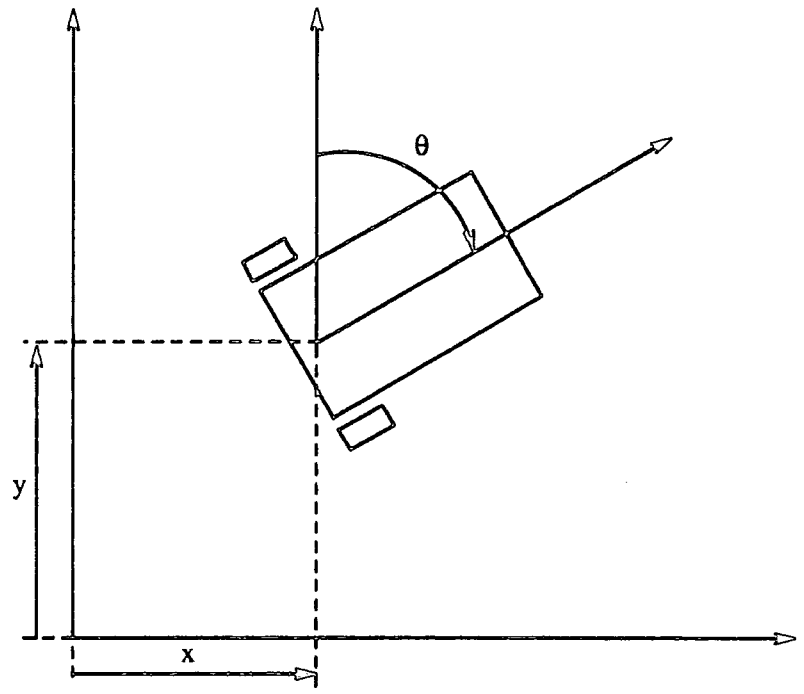

Figure B.1: Definition of x, y and θ for the AGV

B.1 Triangulation

Triangulation techniques involve taking measurements from an unknown position, the location of the vehicle, to known locations, beacons. The measurements can be of angles from the vehicle's heading to a beacon or of the range to a beacon. The system implemented on the vehicle takes only angular measurements to avoid potential interference with any obstacle detection sensors. In the fully generalised case of triangulation there needs to be some form of distinction between beacons in order to associate them with particular measurements. In the case where the system is being used as a correction scheme rather than a stand alone location system beacons can be identified by their position and thus need not be uniquely identifiable.

B.1.1 Triangulation Equations

The equations to locate a vehicle need the angles formed between the beacons from the vehicle's position. Figure B.2 shows these angles as θ_1, θ_2 and θ_3 with the defined positions A, B and C. The beacon positions are fixed and thus the angles and distances between them are known.

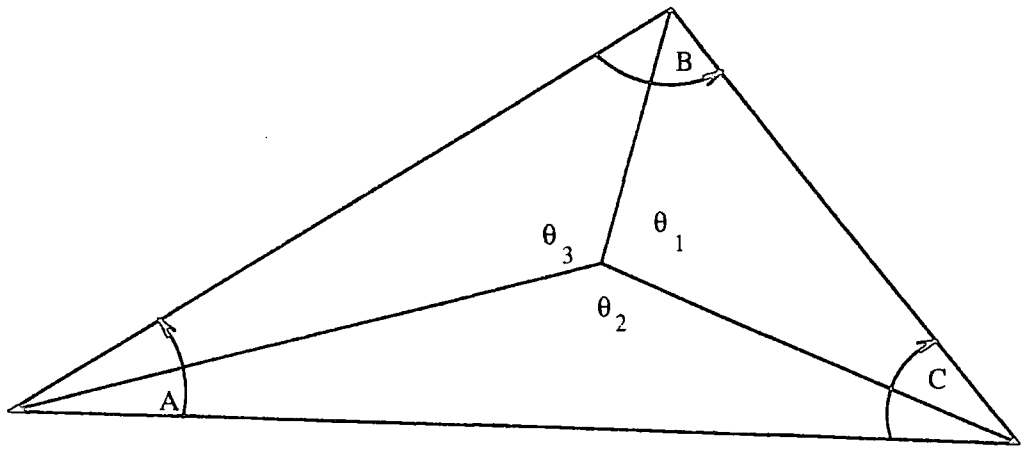

Figure B.2: Angles Used for Triangulation Equations

With beacon positions x_a , y_a , x_b , y_b and x_c , y_c and angles between beacons of $\angle A$, $\angle B$ and $\angle C$ then the vehicle's position x_p , y_p can be found from Equations B.1 and B.2.

$$x_p = \frac{k_1 x_a + k_2 x_b + k_3 x_c}{k_1 + k_2 + k_3} \tag{B.1}$$

$$y_p = \frac{k_1 y_a + k_2 y_b + k_3 y_c}{k_1 + k_2 + k_3} \tag{B.2}$$

where

$$\begin{aligned} 1/k_1 &= \cot A - \cot \theta_1 \\ 1/k_2 &= \cot B - \cot \theta_2 \\ 1/k_3 &= \cot C - \cot \theta_3 \end{aligned} \tag{B.3}$$

Once the position is known any of the measured angles to a beacon can be used to find the vehicle's heading. As can be seen from Figure B.3 the angle from the reference (y axis) to the beacon (ϕ_b) can be found from Equation B.4. The heading angle can then be found by subtracting the measured angle to the beacon (θ_m) from this result as shown in Equation B.5. This result usually requires correction depending on the relative positions of the vehicle and beacon to give an angle in the required range (0 to 2π).

$$\phi_b = \arctan \frac{x_b - x_p}{y_b - y_p} \tag{B.4}$$

$$\theta = \phi_b - \theta_m \tag{B.5}$$

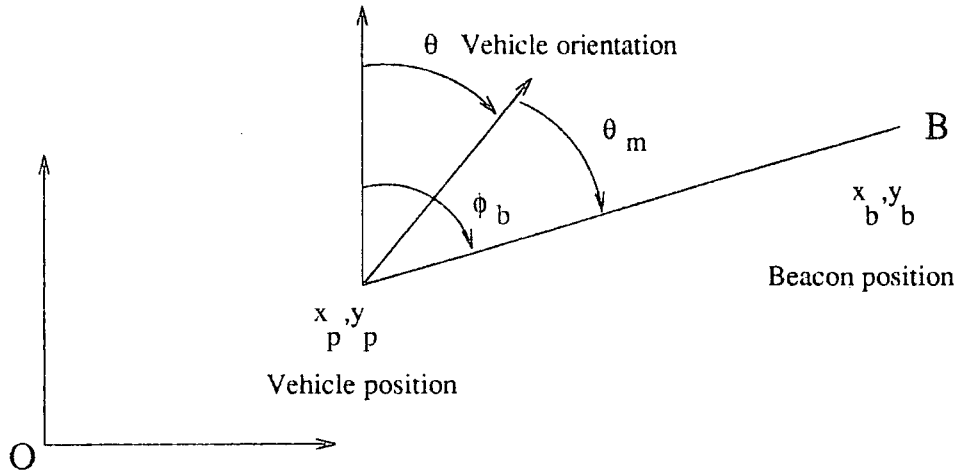

Figure B.3: Calculation of Heading from Position and Beacon Angle

These equations are mathematically correct but the use of the cot function makes the result highly sensitive to errors in the measured angles. This sensitivity was examined using angle measurements containing simulated errors (program *Locerr.c*) and found to result in massive position errors for even a small measurement error.

Consideration was therefore given to alternative techniques which would be able to cope with the inaccuracy of the data.

B.1.2 Triangulation With Inaccurate Data

There is an error associated with each angle measured, at the very least this is the $\frac{1}{4}^{\circ}$ discrete step size of the scanning mechanism and a realistic estimate is around 1°. Figure B.4 shows the effect that measurement inaccuracies have. The resulting three points are all estimates of the true vehicle position and may be distributed around the true position or slightly away from it.

A 'C' program *Locate.c* [58] was used to explore the relationship between errors in measuring angles and the errors in the position estimate. The program calculates the correct beacon angles and then applies an error offset to them and uses these angles to find the three points where the lines from the beacons intersect. An obvious method of combining these values is to average them but by considering the sources of errors a more accurate estimate can be obtained.

The most obvious source of errors comes when the lines from the beacons come close

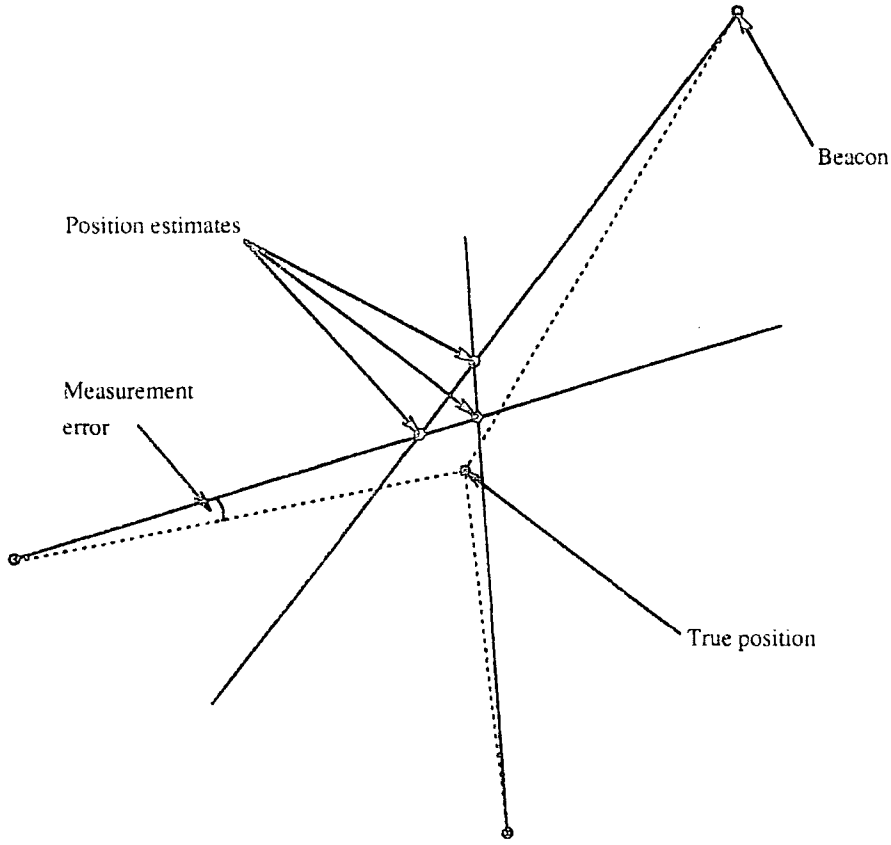

Figure B.4: Points Found When Triangulation Angles Are Inaccurate

to being parallel where a small angular error can give a very large change in the position at which they intersect. Consideration of the vector intersection equation, shown as Equation B.6, demonstrates that the result is very susceptible to variations in the low values of $sin(\theta - \phi)$ as the lines become parallel. As a result this value, the sine of the angle between the lines was thought to be a good weight for the calculation of a weighted average of the three points.

Two vectors of the form $\begin{bmatrix} a \\ b \end{bmatrix} + \alpha \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix}$ and $\begin{bmatrix} c \\ d \end{bmatrix} + \beta \begin{bmatrix} \sin\phi \\ \cos\phi \end{bmatrix}$ intersect at

$$\begin{bmatrix} a \\ b \end{bmatrix} + \left(\frac{(b-d)\sin\phi - (a-c)\cos\phi}{\sin(\theta - \phi)} \right) \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} \qquad \text{(B.6)}$$

A series of experiments using the sine of the angle of incidence or its square as weights on the position estimate was carried out and are summarised in Table B.1. The data was gathered by considering the AGV in eight evenly spaced orientations at 1560 positions in the room. The table shows the maximum distance error in meters from the true position

	Maximum error	Mean of errors	Standard deviation
Raw data	2.463	0.126	0.0278
Average	0.868	0.096	0.0074
Sine weighted	0.268	0.072	0.0016
$Sine^2$ weighted	0.258	0.070	0.0016

Table B.1: Comparison of Triangulation Error Reduction Methods

with a 1° maximum error on any measurement.

This shows that the raw data contains some very large errors with one interception point nearly two and a half meters from the correct position. Applying the averaging techniques yields better results with the average error for 'sine squared' weighting reduced to seven centimeters. The worst result however was still some twenty centimeters from the actual position so even with the corrective weights large errors could be produced.

B.1.3 Position and Orientation Correction

Using the above data it is clear that triangulation with data of limited accuracy is possible but must be carefully checked to avoid large errors. To prevent large errors wrecking the operation of the location system two additional techniques were adopted in the final correction routine. Firstly a correction was rejected if either its x or y estimate was more than ten centimeters from the current position estimate. Secondly a simple filter was used to average the past ten correction values before the odometry process was instructed to correct its position estimate. By rejecting large errors spurious measurements which could not be accounted for by wheel slip or stepper motor stalls are rejected. By averaging successive position corrections the effect of 'bad' positions which produce unreliable data is minimised by ensuring that the final correction has been verified from a number of points as the vehicle moves.

While the angles to three beacons need to be known for triangulation of the position a correction of the vehicle's heading can be made from just one beacon using Equation B.5. In a similar manner to the filtering done on the triangulation data large angle errors (over 5°) were rejected and corrections made only after ten correction values had been averaged. Since triangulation only takes place when all three scanners are tracking a beacon but orientation can be corrected from just one beacon the corrections to orientation tend to happen more frequently. The system operates by sending a correction to orientation or

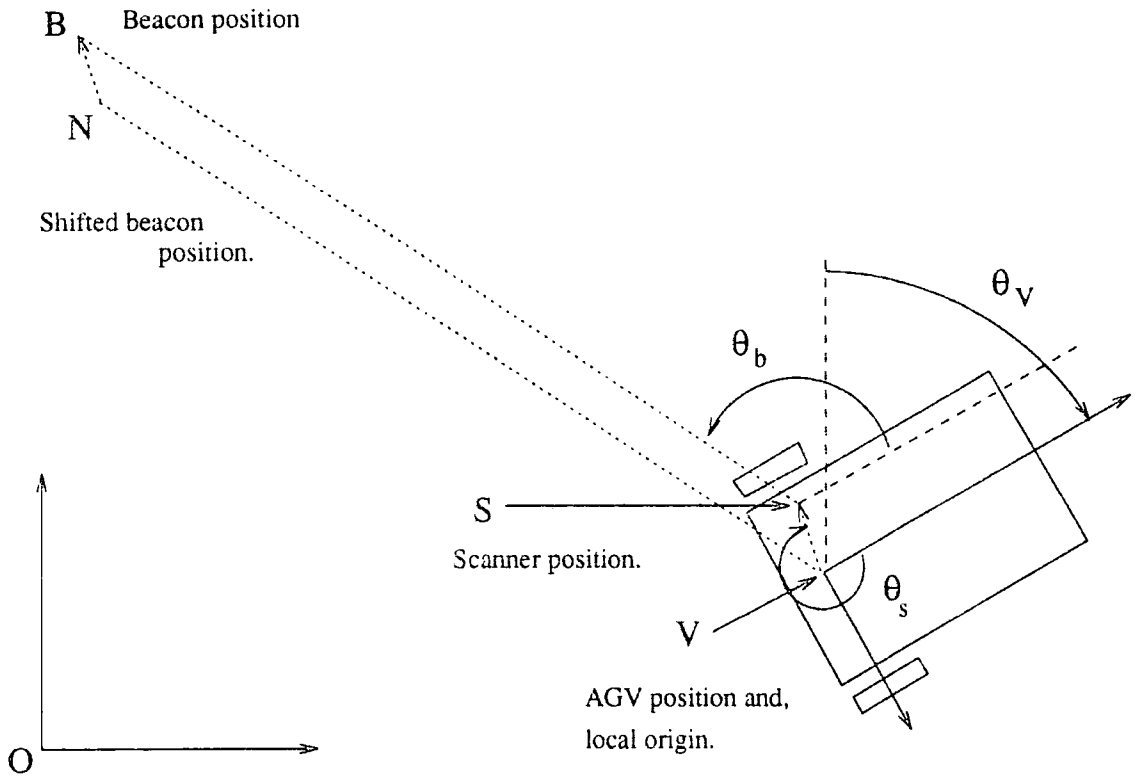

Figure B.5: Illustration of Correction Made for Scanner Position.

position as soon as it has gathered the required ten valid samples.

Before the scanner information can be used for triangulation however it must be converted from the form in which it is gathered, rotation angles of scanners, to give the angles which would define the direction of lines from the beacons to the reference point on the vehicle. The problem with this is that the scanners themselves are not at the reference point of the vehicle and this must be allowed for. This is illustrated in Figure B.5 where B is the beacon position in global coordinates and the scanner at position S on the AGV has measured an angle θ_b to the beacon relative to the local origin. The local origin is at the global position V and has orientation θ_V relative to the global reference. To allow for the fact that the scanner is not at the local origin for the AGV the apparent beacon position is shifted from B to N by the distance that the scanner is from V. The position of the scanner in the local coordinate frame is known, it is a distance s from the origin at an angle θ_s. This converts to a vector of length s in direction $\theta_V + \theta_s$ in the global coordinate scheme which is used to find the apparent beacon position N. The direction of the vector $\vec{NV}$ is $\theta_V + \theta_b$ and it is this vector which is used in the routine to find intersection points. This introduces further inaccuracies since the orientation of the vehicle must be known to

	Maximum	Minimum	Mean	Standard deviation
X error	0.004	-0.048	0.024	0.021
Y error	-0.008	-0.047	-0.028	0.016
Distance error	0.049	0.040	0.045	0.004
Orientation error (degrees)	0.379	-0.585	-0.118	0.288

Table B.2: Results of the Static Tests of Location System

provide the corrections. It is possible that an iterative scheme might reduce errors by using the intersection points to calculate new estimates for the vehicle's orientation but this was not pursued, partly because of time constraints and partly because it was believed it would place too much emphasis on one set of measurements.

To test the effectiveness of the beacon location system and the error reduction measures described above tests were carried out at twenty evenly spaced positions in the laboratory. Results from this are shown in Table B.2 and show that the system positioned the vehicle in the laboratory to within five centimeters of its true position and a degree of its true orientation. In view of the fact that the positioning of the vehicle for these tests using a tape measure can only be regarded as being accurate to a few centimeters and a couple of degrees these are very good results. When the vehicle is in motion the positioning is not so accurate but the position correction system has been shown to provide a useful backup to the odometry system.

B.2 Location System Hardware

The system was designed to operate simply and based on readily available components. The infra red beacons are mounted close to the ceiling so as to be visible as often as possible. The beacons consist of eleven infra red emitting diodes spread out to cover the floor evenly. The receiving diodes are mounted inside aluminium boxes capable of being rotated in both the horizontal and vertical by stepper motors. Three of these steerable receivers are mounted on the top of the vehicle and each tracks a beacon continuously. The tracking is controlled by a simple controller realised in hardware while the task of measuring the angular position of the scanner and of locking on to a beacon in the first place is done by some custom designed EPLD's [1] and the computer system.

[1] Erasable Programmable Logic Devices

The use of three continuously tracking scanners has several advantages over a single scanner moving from beacon to beacon. The angles to beacons are measured at the same time so no correction for the motion of the vehicle needs to be made. The movements made by the scanners are fairly slow so reducing the demands made on the motors and the control can be done by a simple electronic circuit. The problems which are introduced are that the scanners are not at the same point thus adding complexity to the triangulation equations and they may block each others views of the beacons. These problems are addressed at the software level, described in Section B.4 by taking into account the scanners relative positions and by controlling which scanner is assigned to which beacon.

The system hardware is concerned with following beacons and allowing the computer system to drive the scanner when necessary. In can be divided into several subsystems, the scanner body and motors, the analogue infra red receiving electronics and the the digital controlling logic.

B.2.1 Scanner

The scanners themselves are manufactured out of aluminium and consist mainly of two 1.8° stepper motors geared down by a factor of 11:4 rotating a 50x76x250mm box with the receiving diodes in one end and a 15x12mm hole in the other to admit the beacon signal while cutting out most of the ambient light. The rotation about the vertical axis (horizontal scan) can be a full 360° while the vertical scanning is restricted to $+90°$ and $-70°$ from the horizontal. Both axes have reference points marked by optical switches. The bulk of the electronics is on a single board in the base of the scanner while the stepper controllers are mounted adjacent to each motor and the first amplification stage is on the board with the receiving diodes. A diagram of a scanner can be seen in Figure 2.12.

The beacons consist of eleven infra red transmitting diodes mounted on the face of a hemisphere (half a tennis ball) and arranged to evenly cover the floor area the beacons faces. The circuit is simply a 5V power supply connected across the diodes in parallel. Each diode is in series with a 33Ω current limiting resistor.

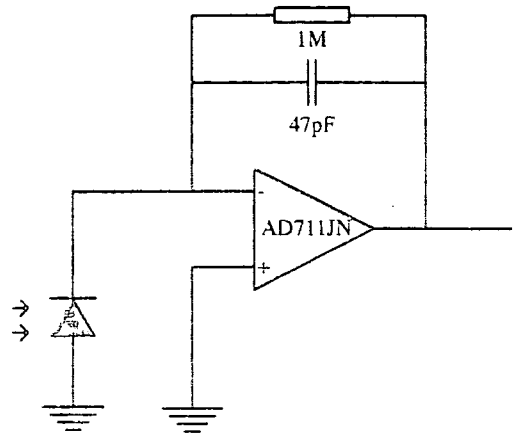

Figure B.6: Infra Red Detection Circuit. One of Five.

B.2.2 Receiver Electronics

In essence the tracking controller is a simple proportional controller applied to each axis of the controller. An integral term cannot be used to control stepper motors as there is always a steady state error associated with them due to the step size. There are five receiving diodes arranged with a central diode and others above, below, left and right of it. The currents caused in the diodes by infra red light are amplified and converted to a voltage signal by a precision operational amplifier. This circuit is shown in Figure B.6.

The signal from the central diode is amplified and used as a reference level by the controller. This level is also fed to a comparator and compared to a preset level corresponding to the normal excitation of the diode. If this level is not exceeded the output of comparator goes high indicating that the scanner has lost the beacon. To prevent multiple switching of this output the circuit has a hysteresis band about the reference level. This circuit can be seen in Figure B.7. The capacitors in parallel with the feedback resistors form a low pass filter to remove high frequency noise from the signal. There is still a small problem with 100Hz noise from the fluorescent lighting but this is not serious unless a scanner points directly at a light.

As can be seen in the circuit diagram Figure B.8 the signals from the diodes to the left and right of the centre are amplified and then fed into a differential amplifier. This produces a signal which is roughly proportional to the error angle of the scanner. The output of this differential amplifier is compared to the reference level generated by the signal received at the central diode. The comparators used can only deal with positive

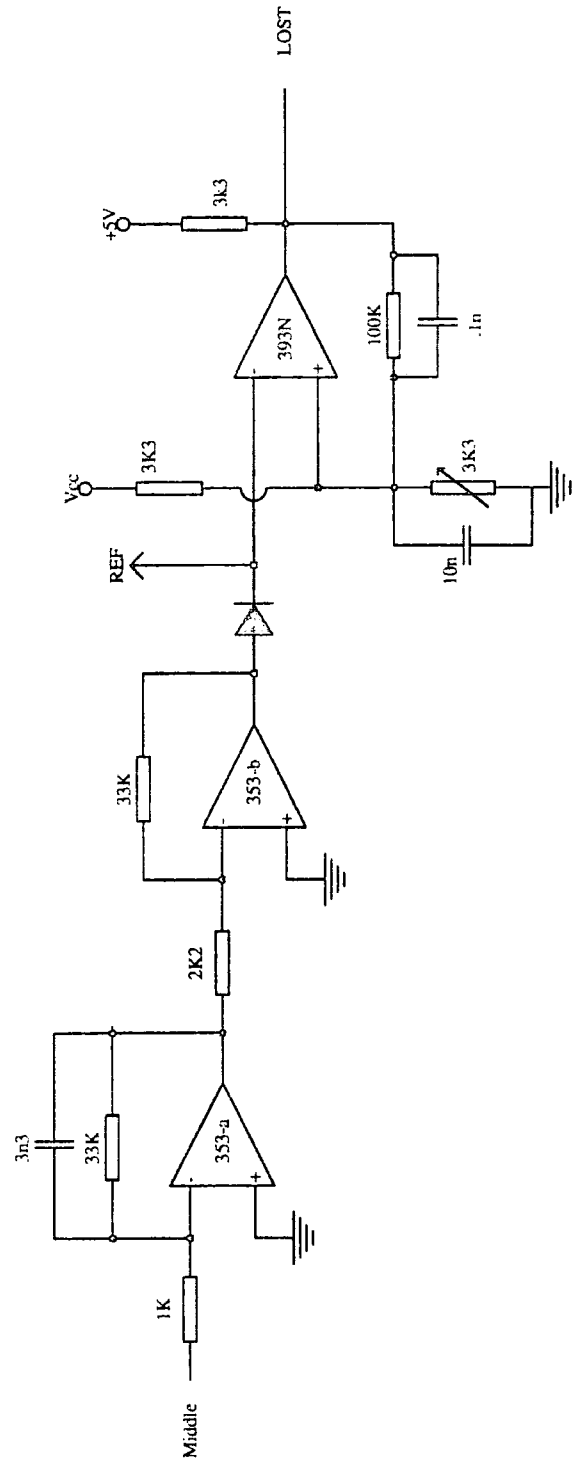

Figure B.7: Generation of Reference Level and 'Lost' signal.

voltage levels so if the output is negative it is inverted before the comparator. Signal diodes prevent the inputs of the comparators ever going negative. If the error signal is above a preset threshold value which corresponds to the error level which a step is required to remove then the corresponding output is sent low indicating that a step is required in that particular direction. The circuit for the vertical error is the same except that the threshold values are set higher since it is not as important to maintain the accuracy of the vertical orientation.

These circuits have been produced on printed circuit boards. The board on which the diodes and first amplification stage are mounted is housed in the back of the movable section of the scanner while the rest of the electronics is mounted in the base of the scanner.

The digital signals produced by these circuits are fed to the boards containing custom designed EPLD's which drive the stepper motors and communicate with the microprocessor system.

B.3 Scanner Controlling Logic

The beacon scanning system was designed to be run by one of the transputers on the AGV. The simplest control task, the continuous tracking of the beacon is handled by a custom designed EPLD [2]. This EPLD also counts the number of steps taken by the stepper motors and communicates with the microprocessor. Complex tasks such as the acquisition of beacons is handled by software and the microprocessor has the power to override the tracking function in order to control the scanners directly. The main reason for this design is that it frees the microprocessor from the most common control function freeing its resources for other, more complex, tasks.

B.3.1 Tracking Function

The EPLD contains the logic necessary to translate the output from the infra red receiving electronics into the direction and step commands for the stepper motor controllers to keep the scanner pointing at the beacon. The basic decoding circuit is very simple and is iden-

[2] Erasable Programmable Logic Device

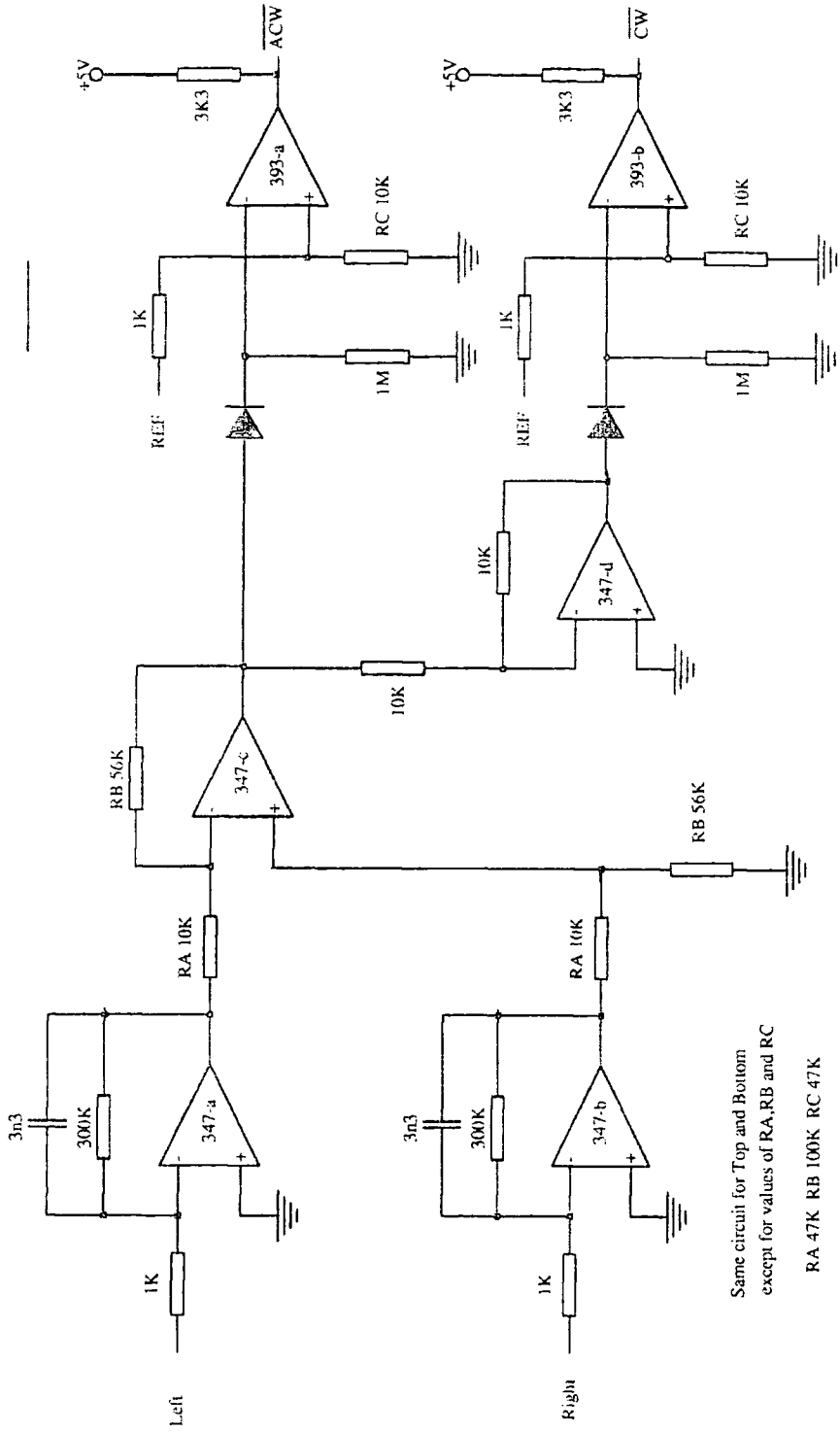

Figure B.8: Generation of Control Signals.

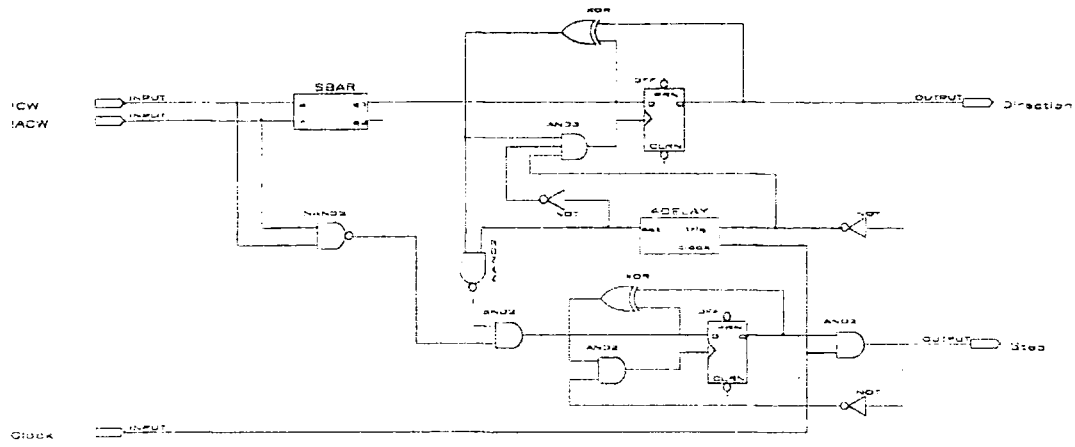

Figure B.9: Logic to Control the Tracking of the Scanners

tical for the control of both the horizontal and vertical orientation. Before the logic was programmed into EPLDs it was tested with discrete components to give a working design.

The output of the receiving electronics indicates whether a clockwise or anti-clockwise rotation is required to lock onto the beacon. If both outputs are high the scanner is pointing directly at the beacon. The controller operates by enabling or disabling a clock signal which drives the stepper motors. The steps are enabled by passing the two inputs through a NAND gate. This simple circuit required modification for two reasons. Firstly the counter which counted the steps would only accept a change in direction when its clock input was low. Secondly the scanner tended to become locked in a series of oscillations about the equilibrium point caused by a combination of the physical inertia of the scanner and the time delay in the receiving electronics.

These problems were overcome by including additional logic which prevented a change in direction from occurring when the output clocking signal was high and prevented the buildup of oscillations by inhibiting additional output for a short period after an overshoot was detected in order to allow the system to settle. The inclusion of the 10KHz clock to allow the use of synchronous logic prevented logic glitches from occurring. A circuit diagram for the tracking controller can be seen in Figure B.9.

To ensure that a change of direction could be applied most of the time the clock was set to have a mark space ratio of 1:99 so that for at least 99% of the time the step output would be low. This was later modified to 3:97 to allow time for the optoisolators to switch the clock signal. The time delay to allow the system to settle was found by observation

to be around 40ms this was set by using an astable circuit. Asynchronous logic however cannot be realised on an EPLD so a method had to be devised to achieve the delay by using synchronous logic linked to the stepper clock.

During testing the clock signal was set at 100HZ. This allows for a maximum tracking speed of 24.75 degrees per second which is reasonable for the AGV. Higher frequencies make the oscillation problem much worse. The required mark-space ratio for the clock was achieved by using a 10KHz clock connected to a counter programmed in the EPLD. This clock setting enables the delay of 40ms to be achieved by simply counting four clock pulses and setting an output high once these have been counted. By using the step output to reset the counter this circuit was made to mimic the operation of an astable.

B.3.2 Other Functions

The scanner control EPLD also provides three other control modes. These are, direct drive, synchronise and constant scan modes. Direct drive mode simply takes the values for direction and step from the control register and outputs them on the scanner control line. Constant scan mode is similar but it treats the setting of the step bit in the control register as an enable or disable bit for the step clock so the scanner is in constant motion while this bit is high. Synchronise mode is used to take the scanner to position where the slotted opto switches are activated to act as an orientation reference. The scanner is rotated in the direction indicated by the direction bit of the control register until the reference switch is reached.

B.3.3 Event Generation

Several possible events can be generated by the scanner EPLD. A maskable interrupt is provided operating from the 'Lost' signal which can call an event on either edge of this signal depending on the setting of the control register. Events are also called when the reference switches are reached. All events are disabled when the scanner is in direct drive mode since it is assumed that in this mode the computer will be constantly monitoring the status of the scanner and no events will be needed.

In accordance with the requirements of the Event_Request line of the transputer the

Address	Name	Type	Contents
00xxx	Vcount	Read only	Steps from reference, vertical channel
01xxx	Hcount	Read only	Steps from reference, horizontal channel
10xxx	Control	Read Write	Control settings from computer
11xxx	Status	Read only	Current status of scanner

Table B.3: The Scanner EPLD Registers.

event signals are active low. There is also an Event_in input so that all devices connected to the event line of a particular transputer can be daisy chained together. The event flag can be read from the event register at address 0x800018 and cleared by writing zero to the appropriate bit at that address. For the scanners bits 3,4 and 5 of the register are used for scanners A,B and C respectively.

This logic was all programmed into an EPLD using the Altera Hardware Description Language (AHDL). The files used to program the EPLDs can be found in [48]

B.3.4 Computer Interface

The EPLD chips were designed to be controlled from the expansion bus of a transputer card designed by Durham Microprocessor Centre. This gives access to external devices from a transputer and normally has sixteen address lines and eight data lines but this was extended to sixteen data lines for the AGV.

Each scanner has its own EPLD containing the tracking logic, two counters, an address decoder, a status register and a control register. The devices used are EPM130J chips and are mounted on specially designed cards next to the transputer boards in the computer rack on the AGV.

The address decoder allows the transputer to access one of four registers on each chip and each chip has its own address as laid out in Table C.1. The layout of these registers can be seen in Table B.3. The lowest two registers on the chip contain the 10 bit counter values for the vertical and 12 bit counters for the horizontal position of the scanner respectively. The next register is a 12 bit control register (Table C.5) which is the only one which can be written to by the transputer. The fourth register, the status register, contains information on the state of the scanner and is summarised in Table C.6. The bits in the control register control the mode of scanner operation, the type of events enabled and the motion of the

scanner. The first two bits determine whether an event will be generated from the 'Lost' signal and which edge will be used. If the 'Event_edge' bit is low then the trailing edge of the signal is used, this is used to indicate the scanner has found a beacon. If the bit is set however an event will be generated when the scanner loses sight of a beacon.

The two mode bits are used to set the scanner into one of the four modes described previously. The scanner channels are controlled by the following eight bits. The 'clr' bits set their associated step counter to 0. The 'ena' bits drive the stepper enable lines while the 'dirn' and 'step' bits control the motion of the scanner depending on the mode selected. Bit four of the status register is the 'Lost' indicator from the receiver electronics signifying that the scanner has lost sight of the beacon. The 'CW' and 'ACW' bits are the outputs of the infra red receiving electronics. These are included to allow for total computer control if desired. The 'sync' bits indicate whether the optical reference switch has been activated while the 'sync_dirn' bits indicate which direction the scanner was travelling when the switch was reached. The direction bits are needed since it takes several steps to cross the region where the switch is activated.

B.4 Location System Software

The interconnection of the software processes which operate the location system and drive the scanners has been described in Section 2.6.2 and shown in Figure 2.13. Associated with each scanner there are four processes which govern its behaviour. The location system control process supervises the system, assigning scanners to beacons and periodically polling the scanner control processes to discover the angles they are measuring to their beacons which is passed on to the correction process whose function has been described in Section B.1.3. The following sections describe how the individual processes operate to provide this information while the code itself is contained in a separate report [49].

B.4.1 Location Control Process

The location control process can be found in the file `location.c` in the `locate` subdirectory. It includes several subroutines in addition to the main routine. These subroutines are *Assign* and *Membership* which deal with assigning the best available beacon to a scanner

Poll_scanners which collects the angle information from the scanner control processes and *Get_beacon_info* which holds the information about beacon positions. The main process initialises the scanner control processes and ensures that all scanners have a beacon to start with before entering the main control loop. This is activated every 200ms and polls the scanner control processes to get their current status, passes the angles measured to the correction routine and then assigns new beacons to any scanners which require it.

The initialisation of the process is started when it receives the start location, sent via the odometry process. The process then sends out the details on beacon positions to the scanner control processes and receives back the vectors defining the positions of each scanner relative to the local origin of the AGV. It relays this information along with the beacon positions to the correction process. The information of scanner and beacon positions is set at compile time but it is important that there is only one source of the information otherwise it is possible that different processes could end up with conflicting information if it were altered. The location system control process then sends a request to each scanner control process to calibrate the scanners. This can cause the scanners to be driven to their reference positions where they activate the slotted optical switches. The best beacon for each scanner is then calculated by using the routine *Assign* which is described in Section B.4.2. The scanner control processes are then polled at 250ms intervals and appropriate action taken depending on the status of the scanner. Once a scanner has been calibrated it is told to locate and track its assigned beacon. If a scanner fails to find the first beacon assigned the next best beacon is assigned until all available beacons have been tried. This set up stage finishes when all scanners are tracking a beacon or a scanner has failed to find any beacons. The success or failure of the starting procedure is communicated down a channel to the AGV controlling process. The vehicle will not start moving until this message has been received.

The location system control process then enters its main loop. It can receive messages from the AGV controlling process (indicating the end of a run) or from the position correction process (indicating that it is ready for new data upon which to base a correction). The beacon assignment must be done centrally to prevent more than one scanner pointing at the same beacon. Equally it is important that the location system controlling process should not become 'blocked', that is waiting for communication from another process, since this would deprive the other process of access to its services. It is therefore set up so that it polls the scanner controlling processes every 250ms and they reply immediately with their

Bit	Name	Meaning
1	NO_BEACON	Scanner is not currently pointing at a beacon.
2	BUSY	Scanner is 'busy' untangling its cables.
3	SYNC	Scanner is trying to drive to its calibration position.
4	CALIBRATED	Scanner calibration is valid.
5	SEARCHING	Scanner is looking for a beacon.
6	FOLLOWING	Scanner is tracking its assigned beacon.
7	SEARCH_FAIL	Scanner has failed to find the assigned beacon.
8	ERROR	Scanner has failed to operate as expected.

Table B.4: Meanings of Status Byte Values

current status. The process also keeps track of whether the correction routine is ready for new data so it cannot become blocked waiting to send the data to it.

When it has polled the scanners the location control process then has a status byte for each scanner and, if the correction process is ready for more data, the angle each scanner is pointing at. Each bit in the status byte conveys information about the scanner as explained in Table B.4 and the controlling process uses this to decide what action, if any, is needed. The bits are given representative names to make the code easier to read, the definitions can be found in the header file `codes.h`. If a scanner indicates that it has failed to find the beacon it has been assigned the system assigns it the best available beacon. This may be the same as the one it previously failed to find so the system allows three attempts on the same beacon before it is marked as 'obscured' in an array so the system will not try it again. Should all available beacons be obscured the array is cleared to allow the system to retry all available beacons in the hope some were just temporarily occluded. The information about the current angle that each scanner is pointing at is passed to the correction routine, along with the beacon number it has been assigned. If the status byte indicates that the scanner is not currently pointing at a beacon then the angle measured is replaced by a flag value (0xfff) to inform the correction scheme that the value should be ignored. The loop runs until the AGV controlling process sends a message forcing shutdown of the system. When this occurs the location system control process shuts down each of the scanner controlling processes when they complete their current task before terminating the event handler, position correction routine and finally itself.

B.4.2 Beacon Assignment

The task of beacon assignment is dealt with by the routine *Assign*. This routine is a simple fuzzy logic expert system which examines the different factors governing the acceptability of beacons and selects the one which is given the highest acceptability value. There are two factors which cause a beacon to be totally rejected, it is assigned to another scanner or is marked as obscured. Three other factors are then used to decide how appropriate a beacon is. The first of these is the range to a beacon, too far and the scanner will not be able to pick up the signal from the beacon, too close and the angle measured to the beacon will change very quickly as the vehicle moves and the scanner may lose track of it. Secondly it is important to consider how far the scanner will have to rotate to acquire the beacon. It takes just under eight seconds for a scanner to rotate through 180° which will deprive the vehicle of information for forty control interrupts so it is important to minimise the rotation needed to change beacons. Finally the view of the scanner is considered, ideally each scanner would look for a beacon away from the vehicle, not over it where there is potential for the other scanners to block its view. Each scanner therefore has its own group of angles relative to the vehicle which it would prefer to use to be guaranteed a clear field of view. This is illustrated in Figure B.10 which shows the centreline of the clear field of view for Scanner B, the rear left scanner. The centrelines of the scanner's fields of view are 0°, −117° and 117° respectively relative to the local origin of the vehicle (straight ahead) and the clear view fields spread 155° to either side for Scanner A and 145° for Scanners B and C.

The range is calculated from the position of the scanner which is passed into the routine. The angle to the beacon (in the local coordinate scheme) is calculated from the relative positions of the scanner and beacon and the vehicle orientation, all of which are passed into the routine. The travel angle required and angle in the field of view are found by comparing the angle to the beacon with the current scanner angle and centreline of the scanner's field of view respectively. The assignment routine uses the function *Membership* to discover the fit values for range, travel angle and view angle according to the set definitions shown in Figure B.11. The routine is very similar to the *fuzzycal* routine but is written so it can operate with sets defined as a series of lines rather than just trapeziums. This allows the dual slope seen in the range set.

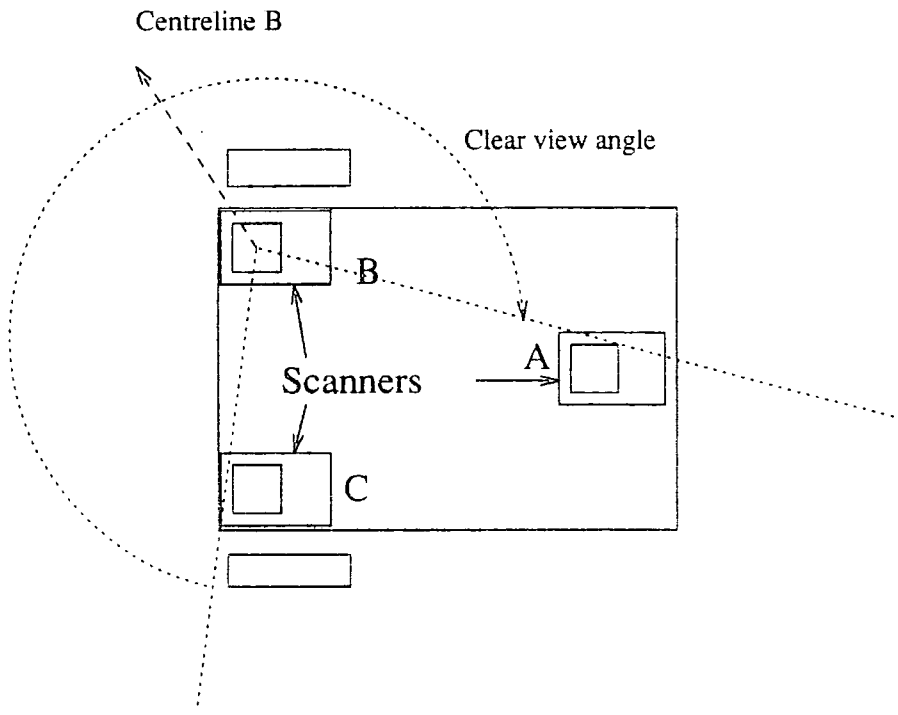

Figure B.10: Illustration of the Field of View of a Scanner.

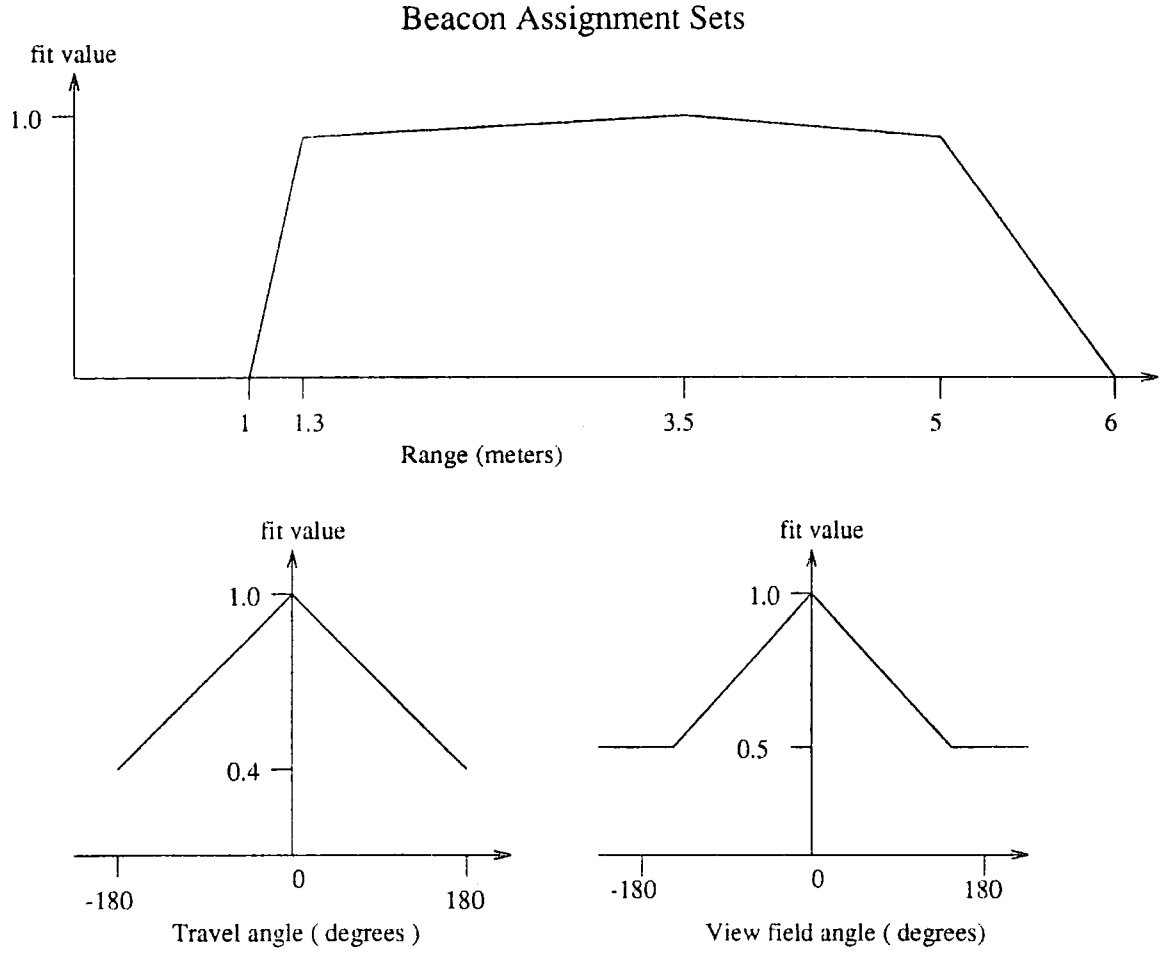

Figure B.11: Set Definitions used in Beacon Assignment.

The overall fit value is found from the minimum of the three fit values. This is the equivalent in fuzzy logic terms of an AND statement, if each of the sets defined in Figure B.11 is taken to define a set named 'GOOD' then the operation can be described as follows.

> Beacon is acceptable if range is 'GOOD' AND view angle is 'GOOD' AND travel angle is 'GOOD'.

The assigned beacon is the beacon number with the highest fit value. If all fit values are zero a beacon number of 254 is returned to indicate there are no unobscured beacons available.

The assignment routine makes a sensible choice as to which beacon should be used. It could be improved by adjusting the travel angle definitions to take into account the cables on the scanners which limit the amount of rotation and can mean that long rotations are needed to untangle them rather than using the shortest distance between two angles.

B.4.3 Scanner Controlling Process

Each scanner has four associated processes. The control process, the driver process, the synchronisation process and a buffer process which handles communication between the synchronisation process and control process.

The control process keeps the time it spends communicating to a minimum so it is always ready to respond to information or requests from other processes. It maintains the status byte described in Section B.4.1 so it knows what the scanner is doing. It passes this and the measured angle to the location control process on request, notes messages from the location system about the assignment of beacons and from the synchronisation process about loss of calibration. It is responsible for monitoring the angle of the scanner and generating a request to untangle the cable if it is needed. Since the scanner driver process is only capable of carrying out one task at a time and cannot answer communications while involved in a task the scanner control process uses the status byte to check that the scanner driver process is ready before sending a new task request. The driver process sends a message to the control process every time it changes status, such as losing the beacon it is tracking. When the angle of the scanner is requested by the location control process the scanner control process reads the counter on the EPLD which contains the number of steps from the reference position the scanner has taken. This is a signed twelve bit number which

the routine converts to a signed sixteen bit one. This value is compared to the maximum and minimum allowed steps to see if the cable is in danger of becoming entangled. The count value is then converted into an angle in radians measured from the local origin of the AGV and passed to the location control process.

B.4.4 Scanner Driver Process

The driver process operates on one of three requests from the controlling process, track a beacon, untangle the cable, and calibrate the scanner. The actual process of tracking a beacon is dealt with in hardware as described in Section B.3 but this software process is responsible for finding the beacon in the first place and coping with momentary loses of the beacon. A calibration request is dealt with relatively simply. If the scanner has not been calibrated before it is assumed to be in a position where clockwise rotations will reach the synchronisation positions and a request is sent to the synchronisation process and the driver process waits for it to reply that the scanner is calibrated. Otherwise the driver process drives the scanner to what it believes is a point ten steps anti-clockwise of the synchronisation positions before sending the request to the synchronisation process. Untangling the cable is done by ordering the scanner to move to a position which is just over half a turn from its present one in the direction opposite to the one which has tangled the cable and then going through the normal procedure to reacquire the beacon. This completes the turn and leaves the scanner pointing at the same beacon as it was before the untangle became necessary.

When the driver process is requested to begin tracking a new beacon it uses the routine *Acquire_beacon* to try and find it using the relative positions of the vehicle and scanner to calculate the required horizontal and vertical rotation angles for the scanner and then uses the infra red receiving electronics to detect the beacon. The horizontal and vertical angles to the beacon are calculated by finding the global position of the scanner from the vehicle's position and orientation and using trigonometry. These angles are converted into desired counts for the number of steps taken, taking the limitations imposed by the cable into account. The scanner is then driven to the required count values and the response from the scanner electronics checked to see if a strong enough infra red signal has been picked up. If not the scanner is driven in a search pattern scanning from plus ten to minus ten counts

in the horizontal and vertical directions to try and pick up a signal from the beacon. If it does the search is ended and the scanner tracks the beacon, otherwise the control process is informed that the beacon has not been found.

The movement of the scanner is controlled by the routine *Move_scanner*. This routine is passed the demand values of the count for the horizontal and vertical stepper motors. The routine reads the current values in the counters and works out the differences needed to reach the required counts and the directions of rotation needed. The routine then writes to the scanner control register on the EPLD to start the scanner moving in the right direction. Since the scanner can take several seconds to reach the required position the process is blocked for the length of time it is predicted the shortest (usually the vertical) rotation will take. The scanner status register is then constantly monitored until the required count is reached, if an overshoot is detected the direction of rotation is reversed. Once the first rotation is complete the process is blocked until the time it is predicted the second rotation will be completed when it is once again monitored until completed. If the flag *beacon_check* is passed into the routine with a non-zero value then the routine also looks for the detection of an infra red source during the motion. This is done by enabling the 'lost' interrupts so that the event handler will pass a message down the 'lost event' channel if a source of high enough intensity is detected. If a message is detected on this channel the scanner is stopped and the status register checked after a delay of ten milliseconds to confirm the detection. If the beacon presence is confirmed the control register is written to to put the scanner into the tracking mode and the routine exited, otherwise the motion continues.

The other main function of the driver process is coping with momentary losses of the beacon. While the scanner is tracking the control register is set so that the loss of a strong infra red signal causes an event. The event handler process sends a message to the driver process when it gets a 'lost' event for its scanner. This causes the driver process to be alerted and it enters 'lost state 1'. In this state the driver process masks the lost interrupts and then waits for a second to see if the beacon is found again. This is to allow for momentary loss of a beacon, caused for instance by someone walking in front of the scanner. If after this time the scanner status register still indicates the EPLD logic was unable to reacquire the beacon the driver process enters 'lost state 2' and tries to reacquire the beacon by going through the same steps it uses when initially acquiring a beacon. Should it fail to acquire the beacon then it enters 'lost state 3' where the scanner is shut down and the controlling

process informed that the beacon has been lost.

B.4.5 Synchronisation Process

This process is concerned with keeping the counts maintained by the counters on the scanner controlling EPLD synchronised with the actual number of steps made by the scanner. These may differ if the stepper motors stall for any reason so the slotted opto switches on the scanner provide a reference point where this may be checked. When either the vertical or horizontal rotation of the scanner activates one of these switches an event is caused. The event handler passes a message on to the synchronisation process to inform it of this. The synchronisation process checks the scanner status register to see which rotation has caused the event and then reads its counter. If the scanner is traveling in the direction in which the original calibration was made the counts must be within one step of zero, otherwise ten steps are allowed to account for the fact that the slotted opto switch is more than one step wide. Should the count be found to be outside the allowed band a message is sent down a buffered channel to the controlling process indicating that calibration is needed. The message is only sent once and subsequent detections of the events ignored until calibration has occurred. Doing this and using the buffered channel ensures that the process is always ready to receive messages from the event handler which it is vital not to block by communication delays.

When a request for calibration is received the process writes to the scanner control register to start this process going. It is handled automatically by the EPLD and all the process needs to do is wait for two events, indicating the scanner is at the synchronisation position and then clear the counter values to zero. When this has occurred it sends a message to the driver process indicating that calibration is complete.

Appendix C

Memory Mapped I/O details

This appendix contains details of the memory mapped I/O programmed into EPLD chips connected to the expansion bus of the transputers. The addresses are defined as C pre-processor definitions in the file `address.h` in the `headers` directory. The following table indicates which addresses different registers are mapped to.

Address	Chip	Name	Type	Notes
00800000	TEST (unused)	register1	Read Write	16 bit register
00800004		register2	Read Write	16 bit register
00800008		register3	Read Write	16 bit register
0080000c		register4	Read Write	16 bit register
00800010	Odometry	Event timer	Read Write	16 bit latch
00800014		-	-	-
00800018		Event set	Read Write	16 bit distributed register
0080001c		Timer control	Read Write	3 bit register
00800020	Motion	Right Latch	Read Write	16 bit latch
00800024		Left Latch	Read Write	16 bit latch
00800028		Control	Read Write	10 bit register
0080002c		Control	-	double mapped
00800030	Odometry	Left count	Read only	16 bit counter
00800034		Right count	Read only	16 bit counter
00800038		Reset	Read Write	1 bit register
0080003c		Reset	-	double mapped
00800040	Scan2A	Vcount	Read only	10 bit counter
00800044		Hcount	Read only	12 bit counter
00800048		Control	Read Write	12 bit register
0080004c		Status	Read only	12 bit register
00800050	Scan2B	Vcount	Read only	10 bit counter
00800054		Hcount	Read only	12 bit counter
00800058		Control	Read Write	12 bit register
0080005c		Status	Read only	12 bit register
00800060	Scan2C	Vcount	Read only	10 bit counter
00800064		Hcount	Read only	12 bit counter
00800068		Control	Read Write	12 bit register
0080006c		Status	Read only	12 bit register

Table C.1: The Root Transputer EPLD Registers.

C.1 Register Contents

The bit settings for the larger registers are as follows. Constants are also defined for the bit settings in these registers in the header file `settings.h` which can be found in [49].

Bit	Name	Function
1	Load	Loads the timer with the contents of its latch
2	reset	resets the timer to 0
3	Event_enable	Enable the timer event
4	-	-

Table C.2: The Event Timer Control Register.

Bit	Name	Function
1	Timer1	Event timer has reached zero.
2	N/C	Unused
3	Lost_A	Scanner A Lost Event set
4	V_sync_A	Scanner A's vertical channel is at reference
5	H_sync_A	Scanner A's horizontal channel is at reference
6	Lost_B	Scanner B Lost Event set
7	V_sync_B	Scanner B's vertical channel is at reference
8	H_sync_B	Scanner B's horizontal channel is at reference
9	Lost_C	Scanner C Lost Event set
10	V_sync_C	Scanner C's vertical channel is at reference
11	H_sync_C	Scanner C's horizontal channel is at reference
12	-	-

Table C.3: The Event Flag Register.

Bit	Name	Function
1	Right Load	Loads the Right counter with contents of its latch
2	Left Load	Loads the Left counter with contents of its latch
3	Right Reset	Resets the Right counter
4	Left Reset	Resets the Left counter
5	Right Direction	Sets the direction of rotation for right stepper
6	Left Direction	Sets the direction of rotation for left stepper
7	Right enable	Enables the right stepper
8	Left enable	Enables the left stepper
9	Buzzer	Sounds the AGV's buzzer
10	Host present	Linked to the front panel switch (READ ONLY)
11	-	-
12	-	-

Table C.4: The Motion Control Register.

Bit	Name	Function
1	Event_enable	Enable events on 'Lost' signal
2	Event_edge	Low for event on trailing edge of 'Lost'
3	Mode0	00 - Direct drive 01 - Constant scan
4	Mode1	10 - Synchronise 11 - Tracking
5	V_clr	Clear V_count
6	V_enable	Enable (switch on) vertical stepper
7	V_dirn	Direction control, vertical (set is up)
8	V_step	Step vertical motor
9	H_clr	Clear H_count
10	H_enable	Enable (switch on) horizontal stepper
11	H_dirn	Direction control, horizontal (set is clockwise)
12	H_step	Step horizontal motor

Table C.5: The Scanner Control Register.

Bit	Name	Indicates
1	N/C	Unused
2	N/C	Unused
3	N/C	Unused
4	Lost	Scanner has lost the beacon
5	V_acw	Vertical anticlockwise tracking needed
6	V_cw	Vertical clockwise tracking needed
7	V_sync	At vertical reference point
8	V_sync_dirn	Direction reference point reached from
9	H_acw	Horizontal anticlockwise tracking needed
10	H_cw	Horizontal clockwise tracking needed
11	H_sync	At horizontal reference point
12	V_sync_dirn	Direction reference point reached from

Table C.6: The Scanner Status Register.

Appendix D

AGV Hardware

D.1 Power Supplies and Isolation

The AGV is powered by two 12V automobile type batteries and there is also a mains supply provided via a long extension cable. A number of power supplies are derived from the batteries via regulators to power the on board electronics. These consist of three 5V, 2A supplies, two bipolar 10V 1A supplies and a 1.25V to 10.5V variable bipolar supply which is currently unused. The stepper driver cards also produce 12V regulated supplies for use by their control signals.

The main driving stepper motors require a 24V unregulated supply and can draw up to 12A. This is supplied directly by the two onboard batteries. The computer system is run from the mains supply by a 5V, 10A power supply. This provides the power for the transputers and EPLDs. The location system is run off the regulated supplies. It requires 5V for logic connections and bipolar 10V for the analogue electronics. The six stepper motors in the scanners each require 24V .5A giving 3A total which can be provided directly from the on board batteries.

A potential problem arises because in order to arrange for a bipolar supply the centre tap of the batteries is connected to the nominal 0V of the computer system. Two of the 5V supplies and the bipolar supplies are derived using this central tap as 0V while the third 5V supply uses the stepper 0V (-12V) as its nominal 0V. This 5V supply is used to send control commands to the scanner stepper motors. In order to keep these supplies separate

optoisolators are used. These provide the necessary isolation between the control signals from the computer and the control signals to the stepper motors. The isolation between the computer system and the main drive steppers are shown in Figure D.1. The isolation for the scanner steppers is done in a similar manner but the outputs of the optoisolators are pulled up to 5V not 12V. Full details about the function of each board in the interface electronics and the connections made to it can be found in [59]

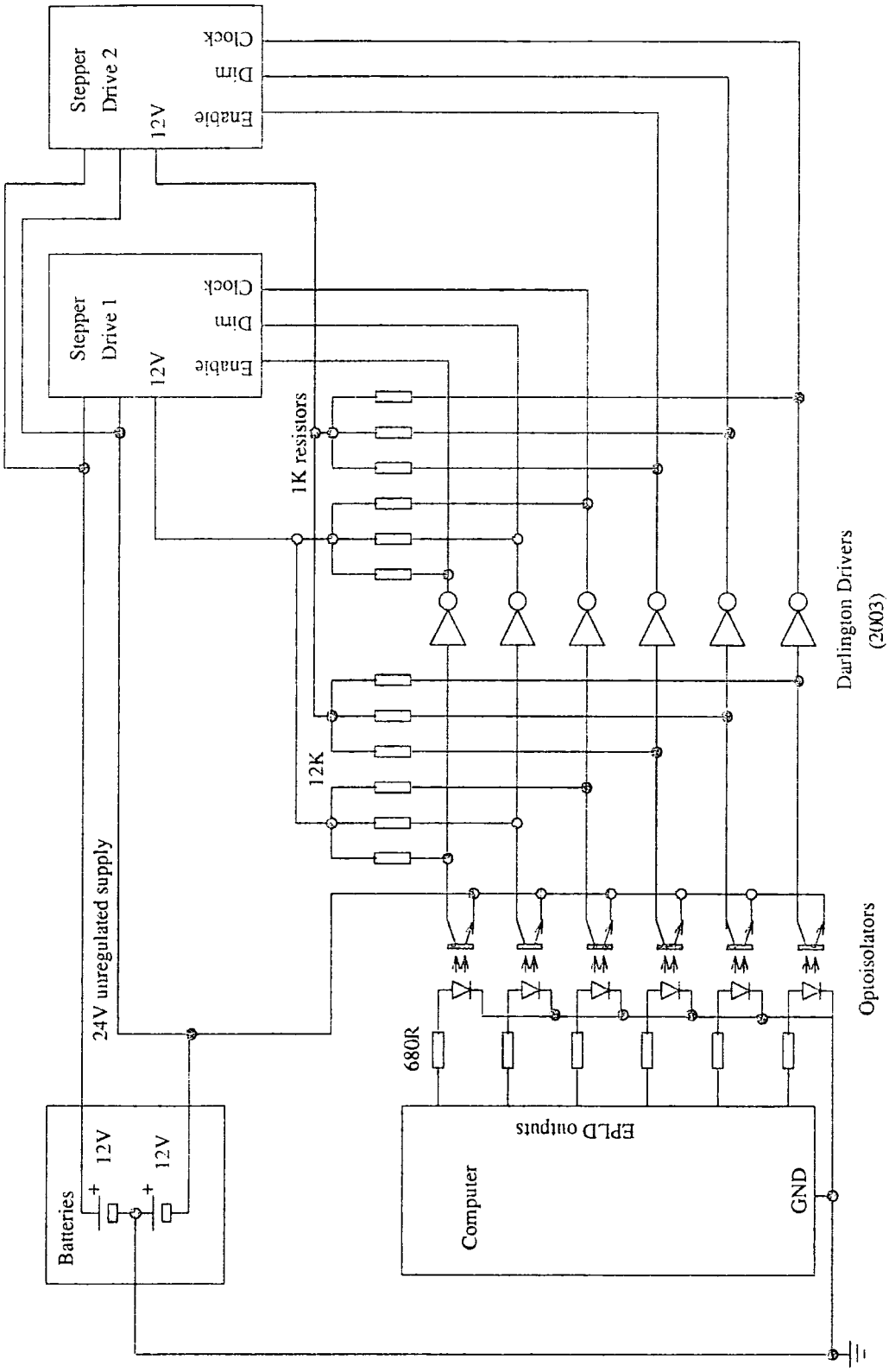

Figure D.1: Main Stepper Driver Isolation Circuit